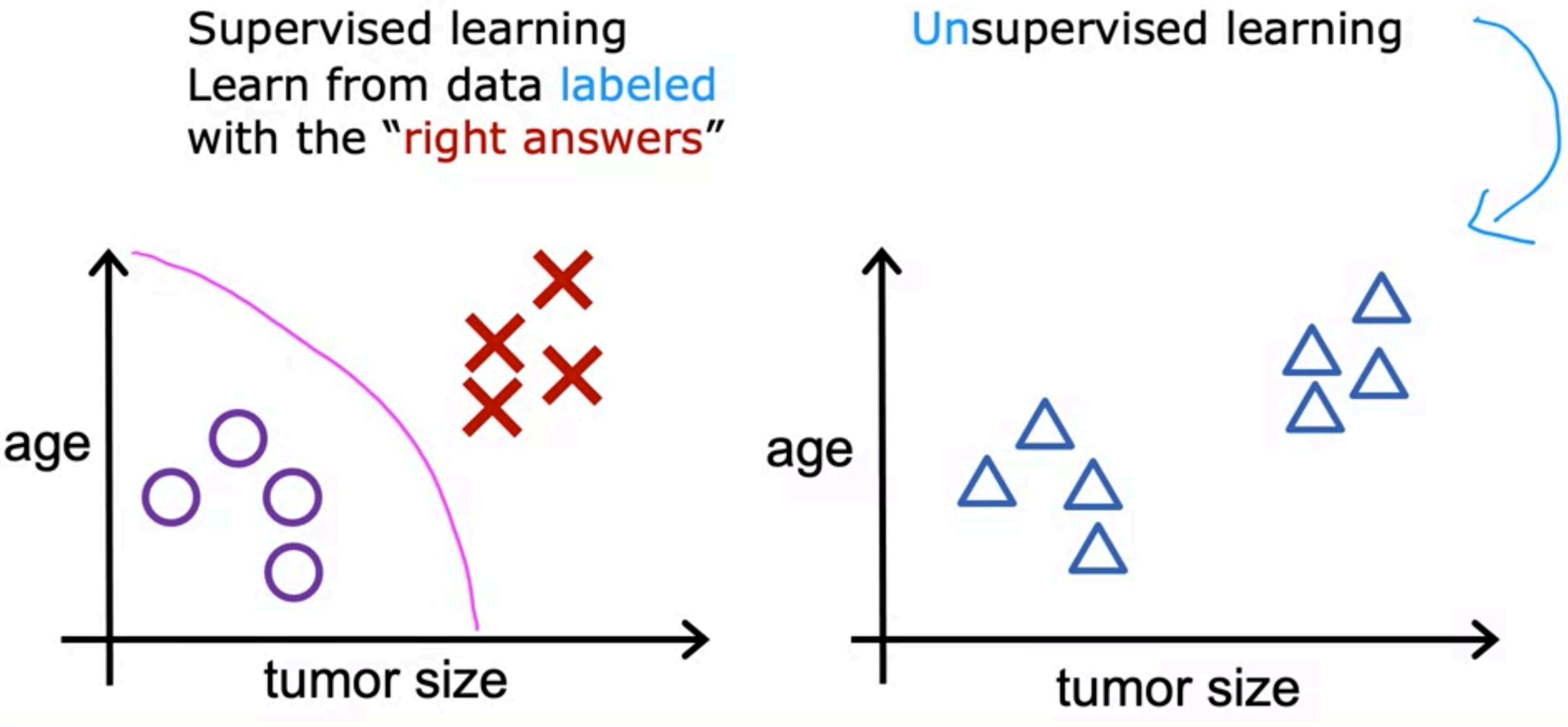
AIBridge

Lecture 8

UNSUPERVISED LEARNING



SU

Credit: Andrew Ng, [Machine Learning]

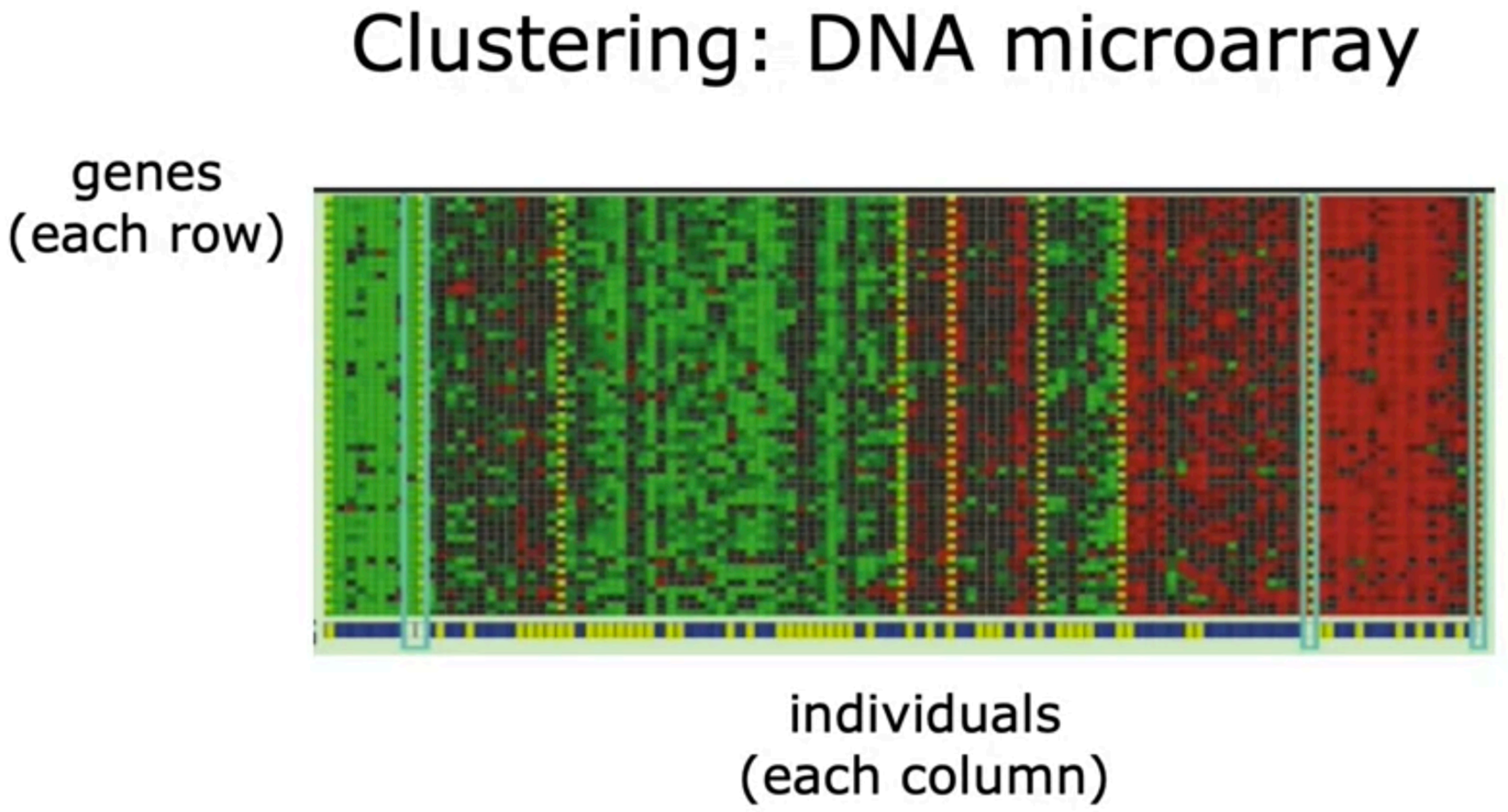
TWO MAIN APPLICATIONS

Clustering

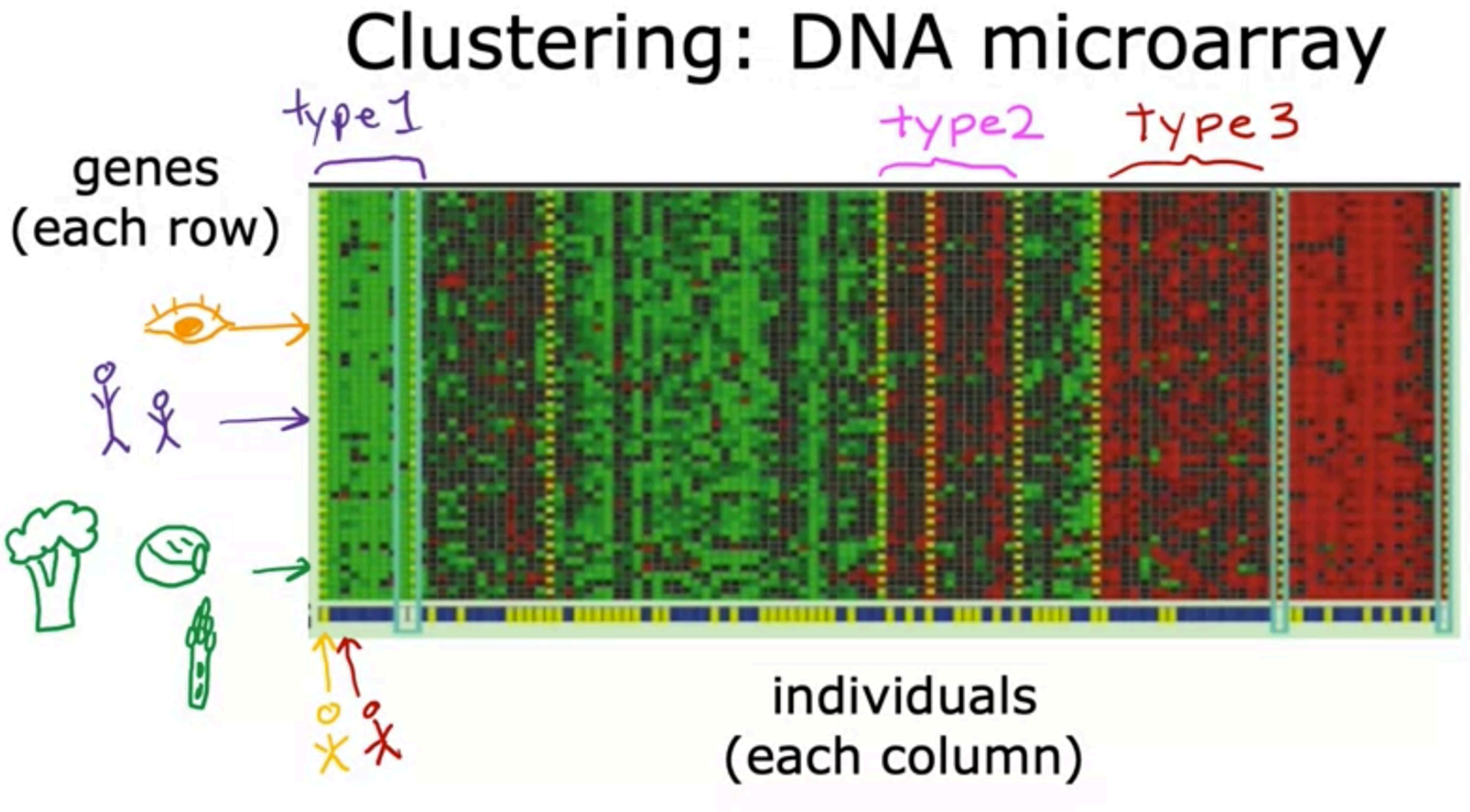
Dimension reduction

Clustering: Google news

Credit: Andrew Ng, Machine Learning

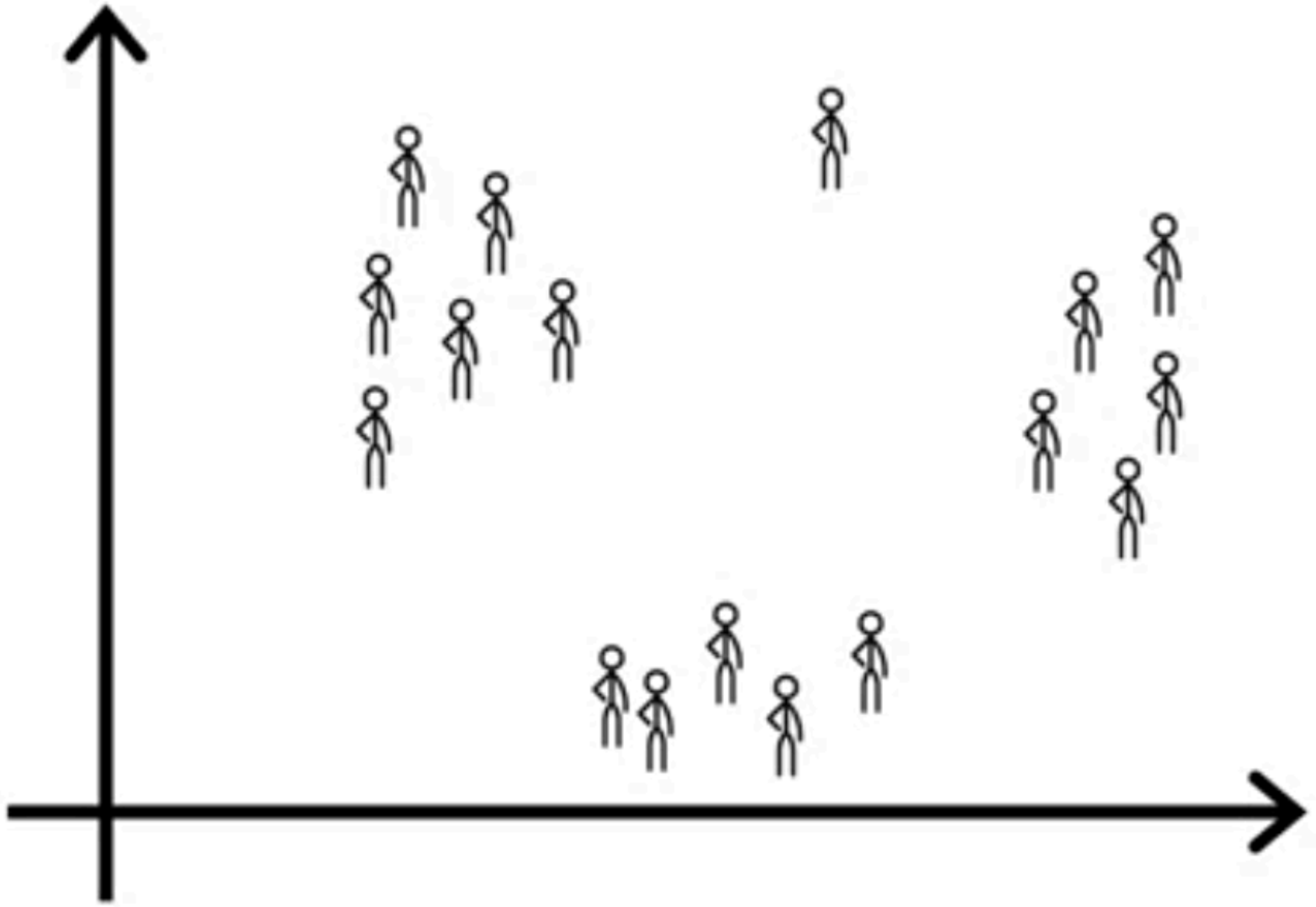


Credit: Andrew Ng, Machine Learning



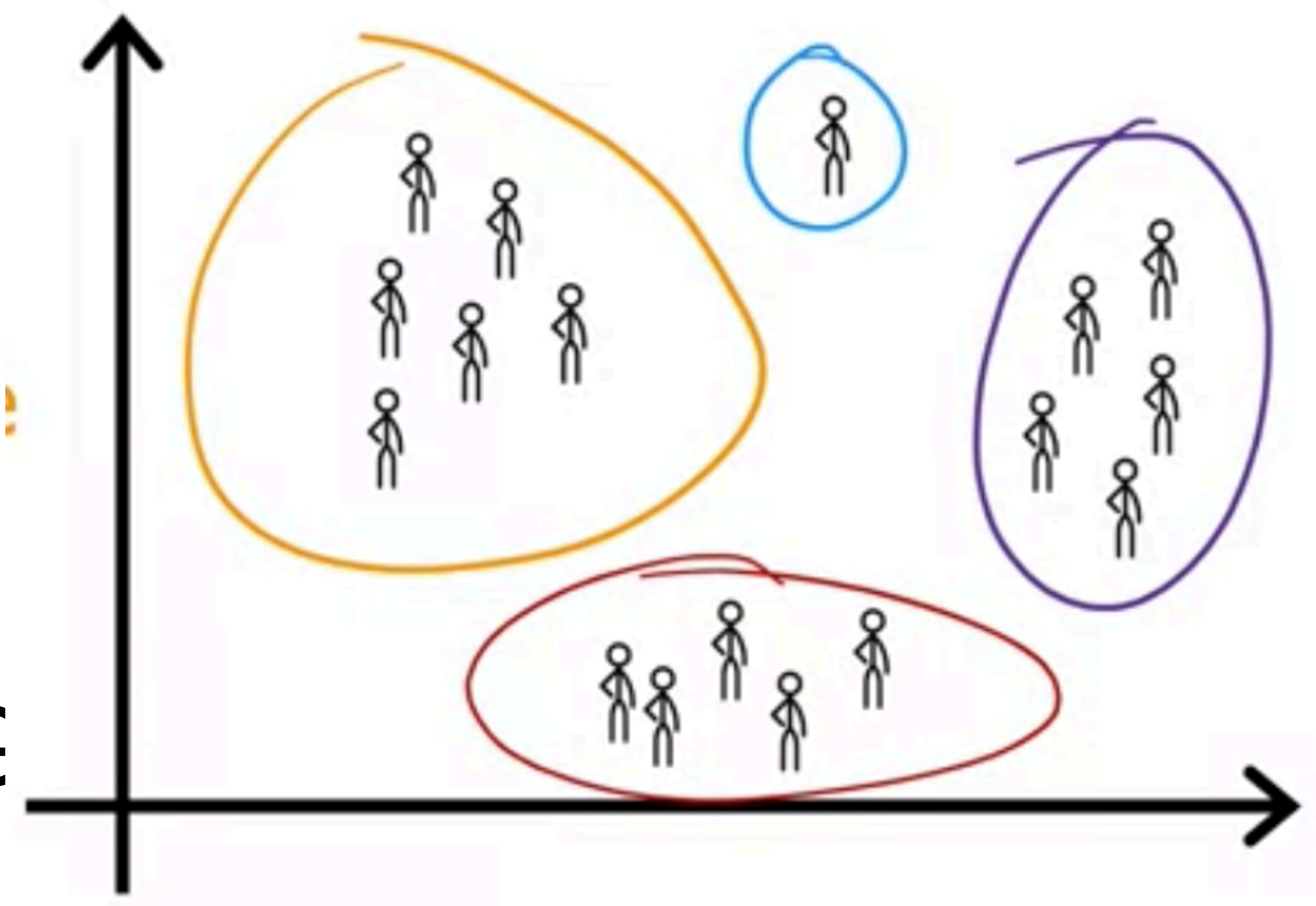
Credit: Andrew Ng, Machine Learning

Clustering: Grouping customers



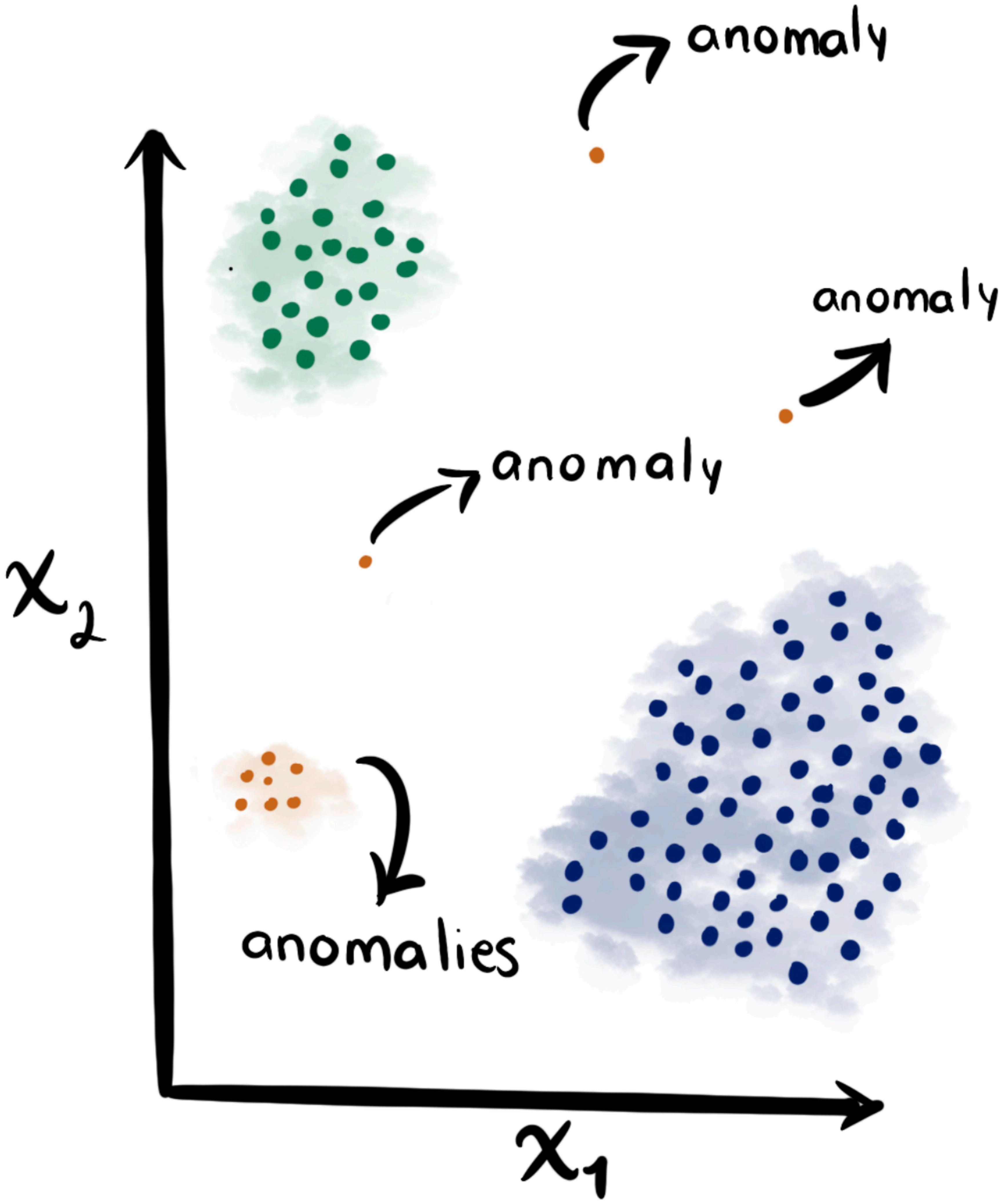
Credit: Andrew Ng, Machine Learning

GROUPING



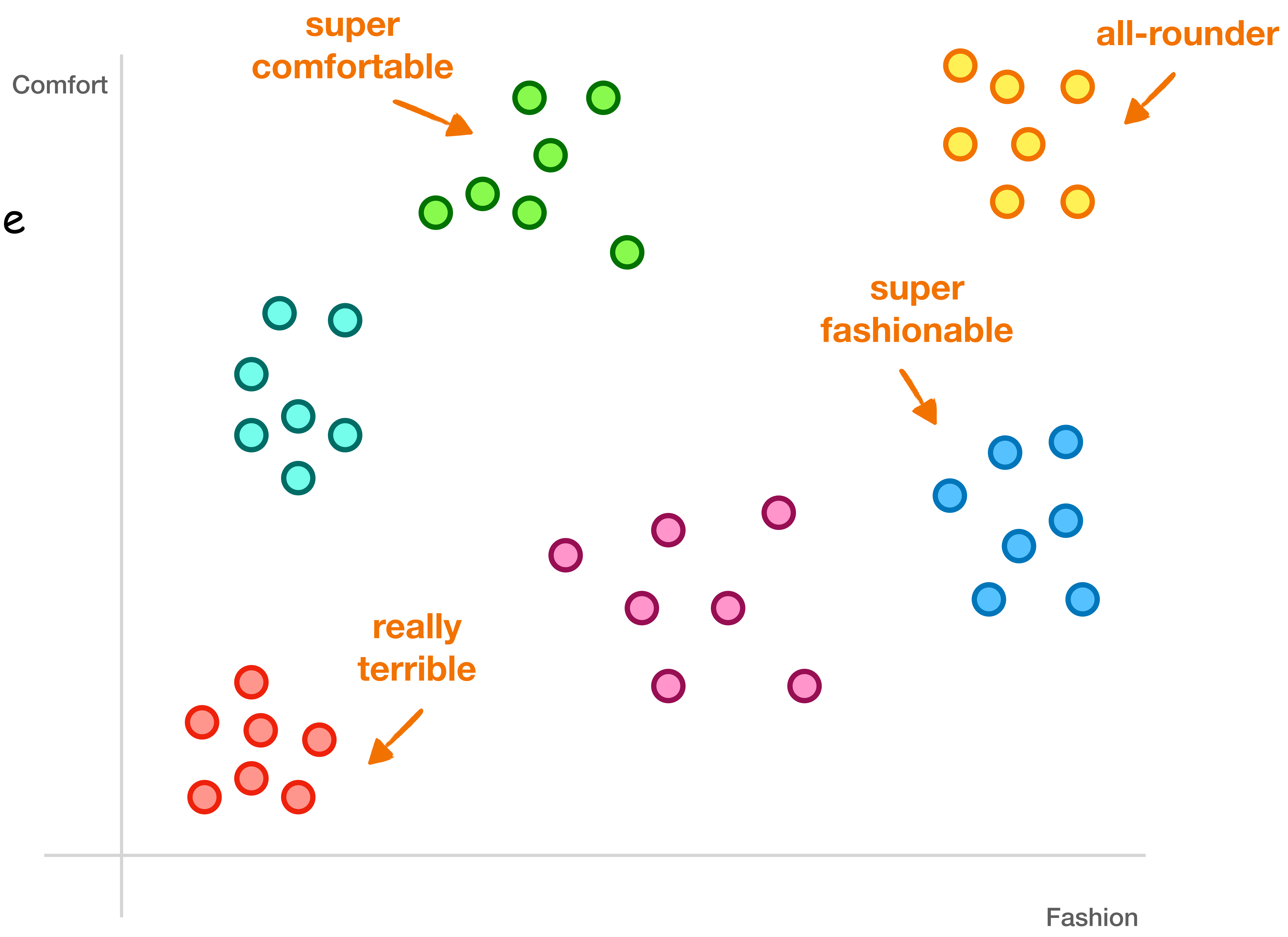
Credit: Andrew Ng, [Machine Learning]

ANOMALY D



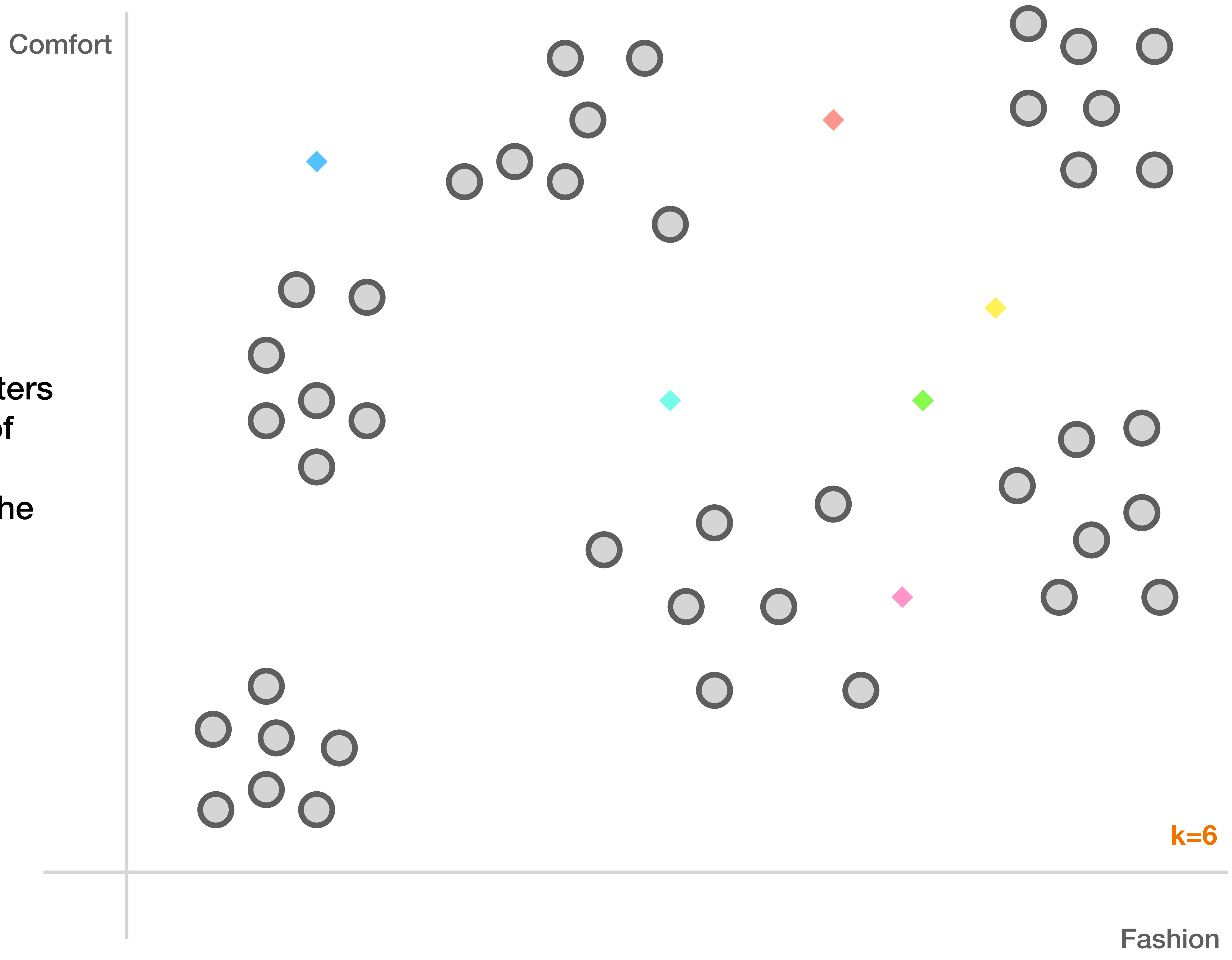
Credit: [Anomaly Detection]

how do we find the clusters?

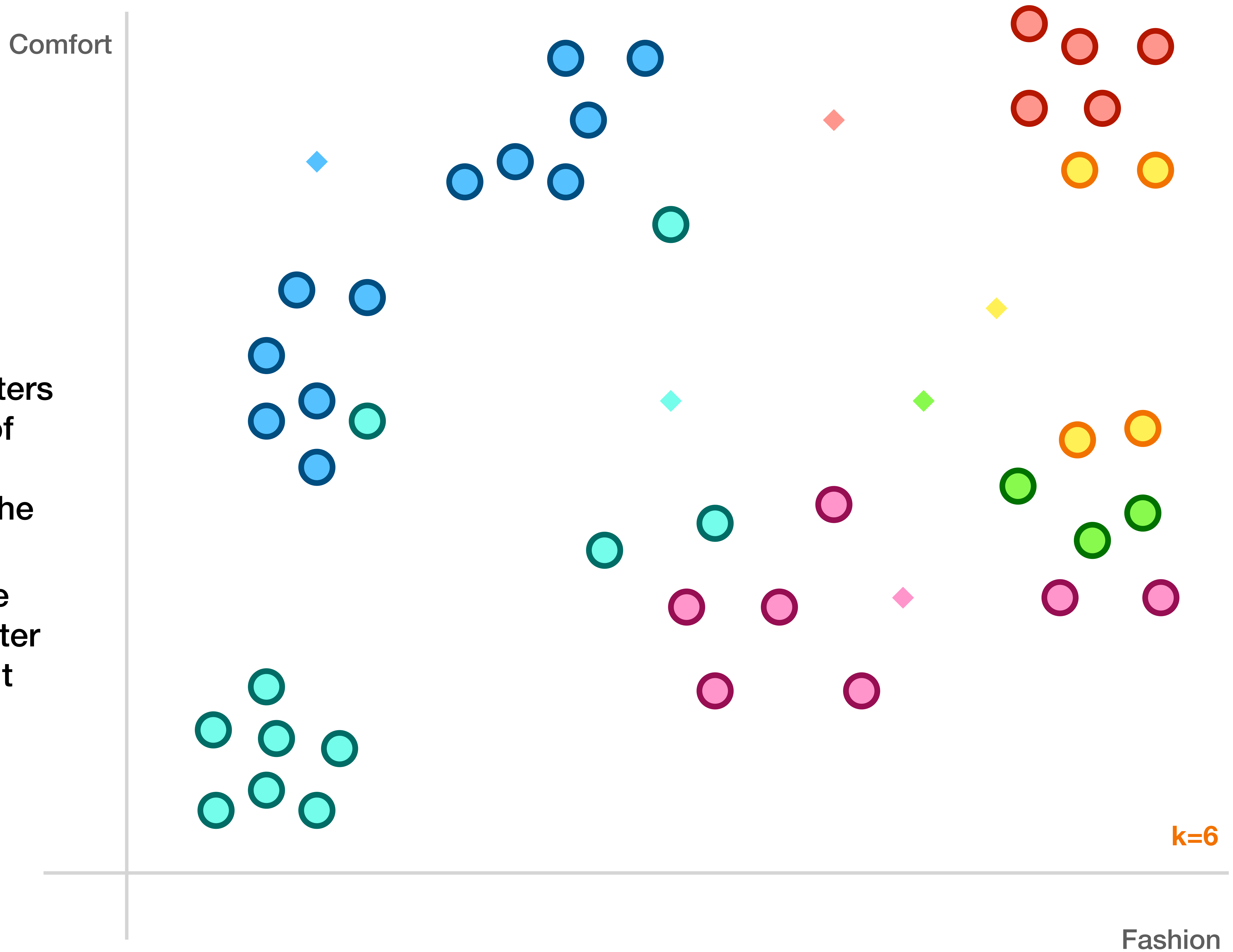


- **clusters** can tell us specifics about the relationship of data
- …even if they are unlabeled! → unsupervised learning!

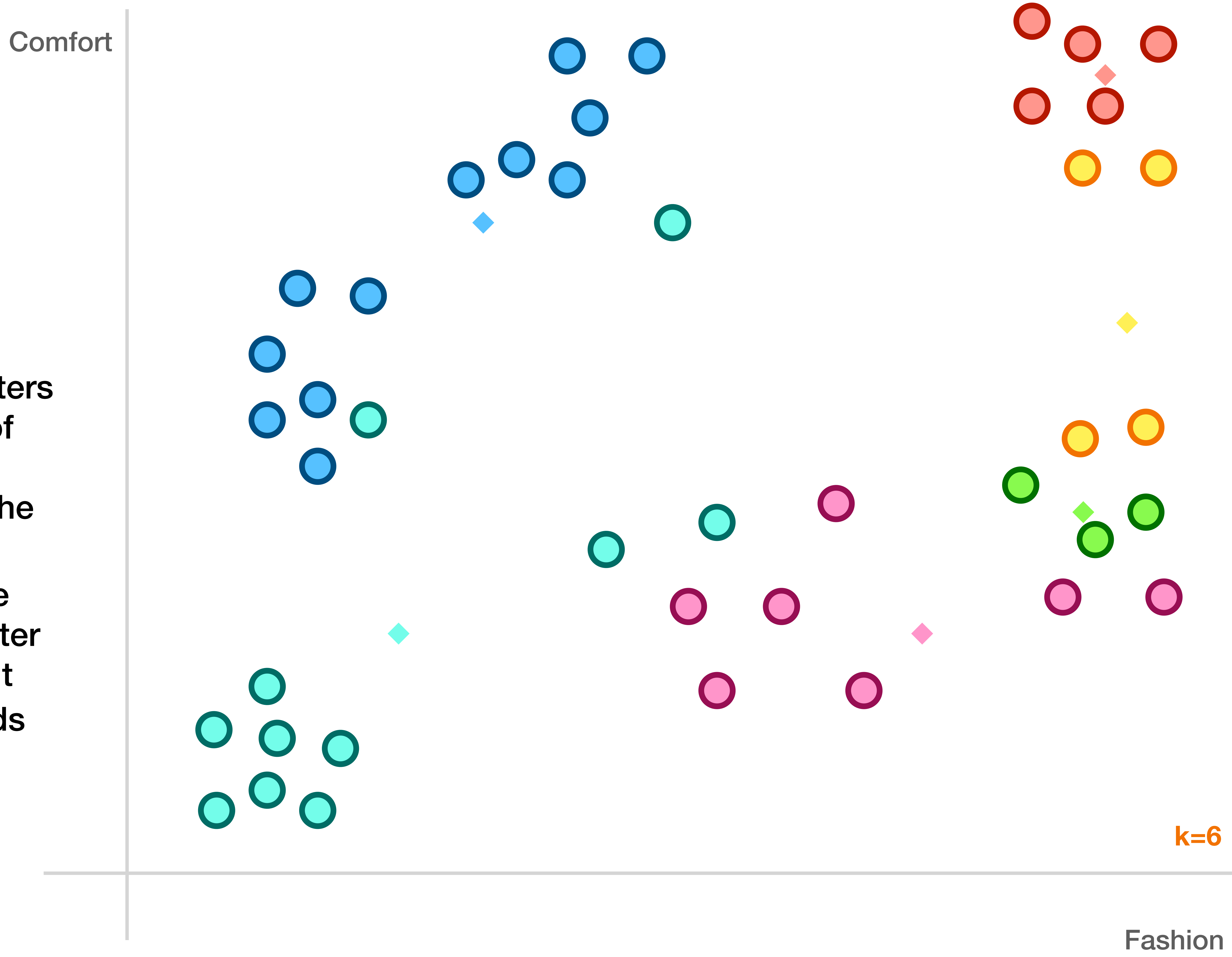
1. pick a K-number of clusters
2. randomly pick a series of **“centroids”**
3. assign each particle to the **centroid** closest to it



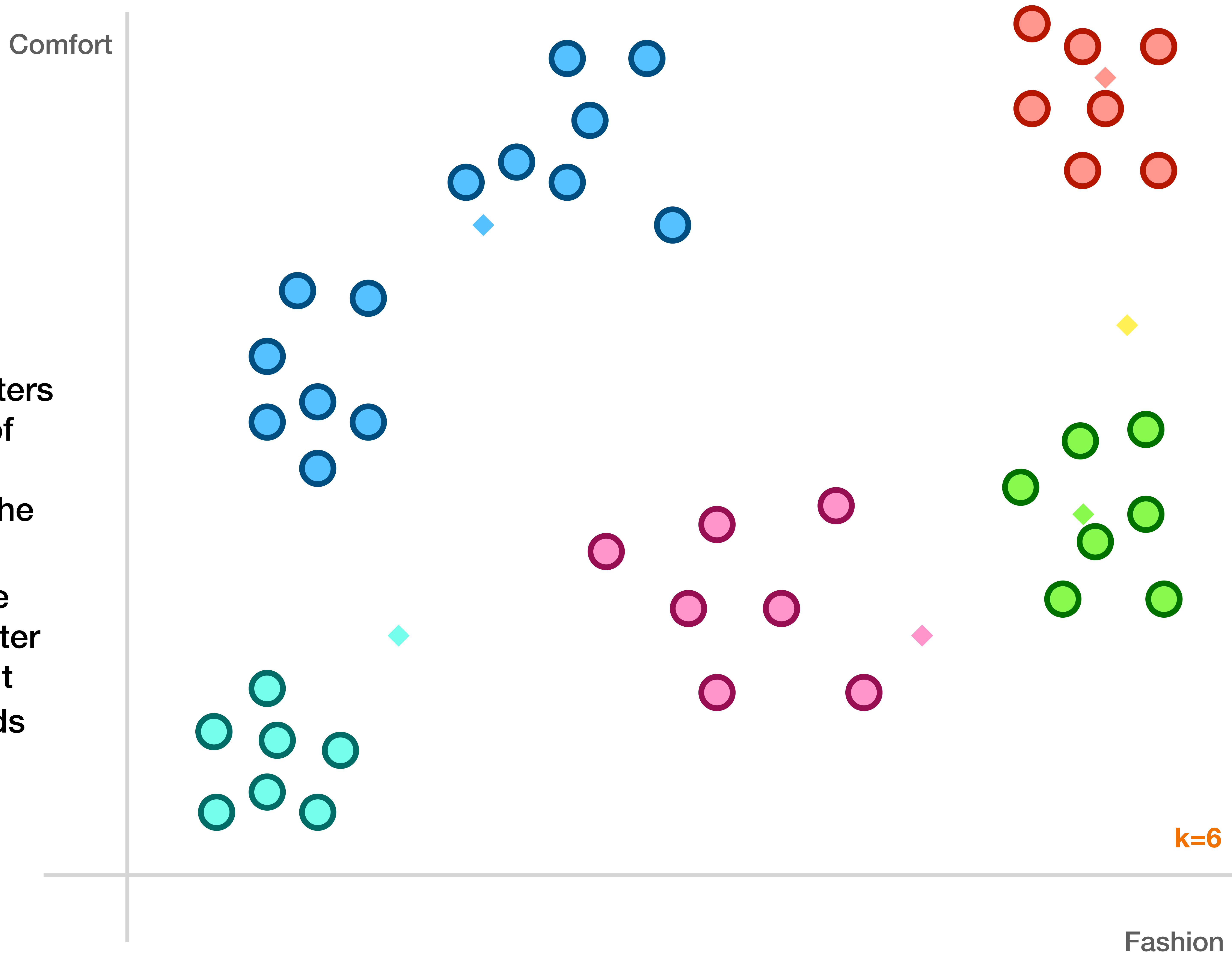
1. pick a K-number of clusters
2. randomly pick a series of **“centroids”**
3. assign each particle to the **centroid** closest to it
4. move the **centroid** to the weighted geometric center of samples assigned to it



1. pick a K-number of clusters
2. randomly pick a series of **“centroids”**
3. assign each particle to the **centroid** closest to it
4. move the **centroid** to the weighted geometric center of samples assigned to it
5. Repeat 3-4 until centroids stop moving!

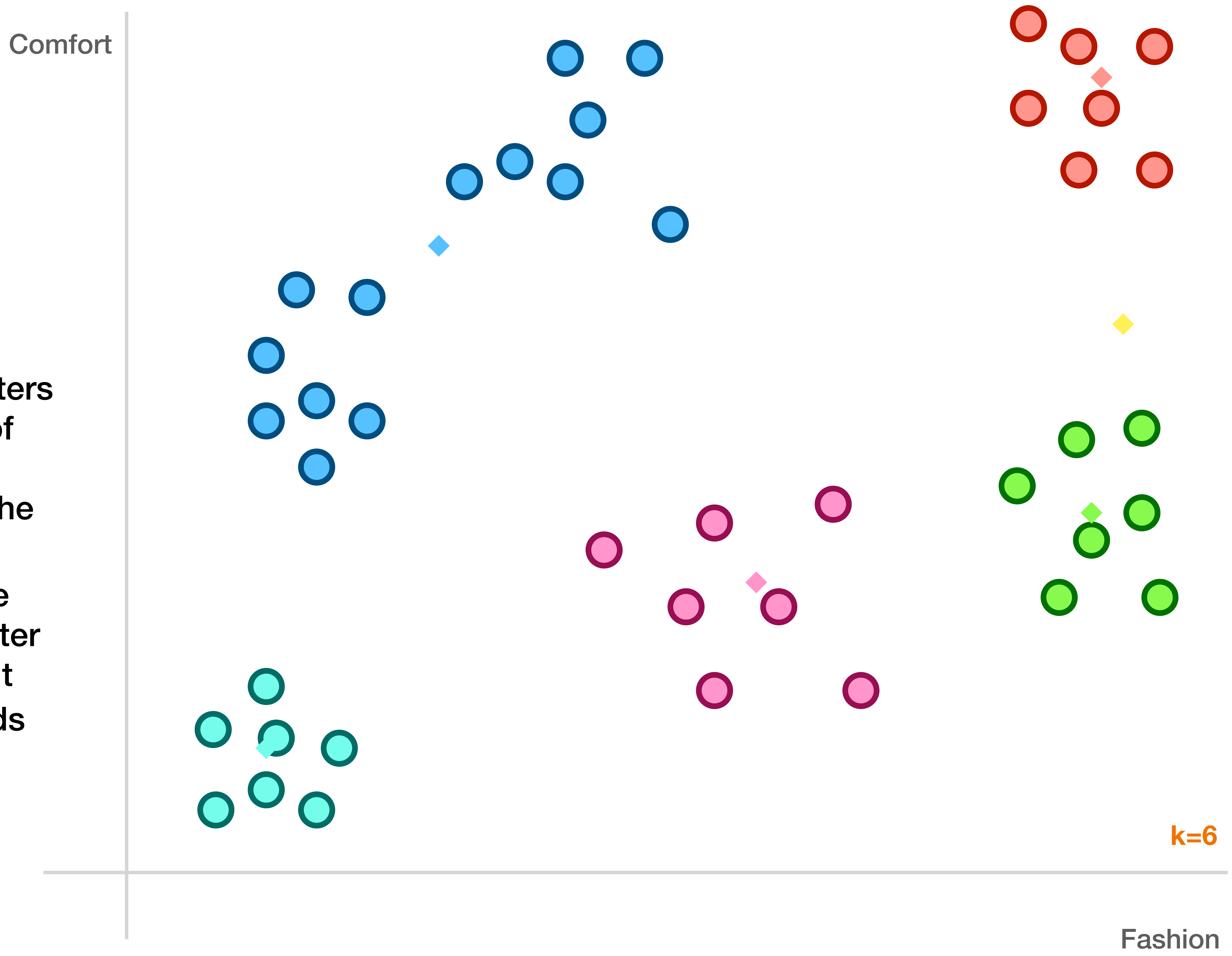


1. pick a K-number of clusters
2. randomly pick a series of **“centroids”**
3. assign each particle to the **centroid** closest to it
4. move the **centroid** to the weighted geometric center of samples assigned to it
5. Repeat 3-4 until centroids stop moving!



k-means clustering

1. pick a K-number of clusters
2. randomly pick a series of **“centroids”**
3. assign each particle to the **centroid** closest to it
4. move the **centroid** to the weighted geometric center of samples assigned to it
5. Repeat 3-4 until centroids stop moving!



Did we get back the same clusters?

Nope. And that’s OK.

Did we get back the same clusters?

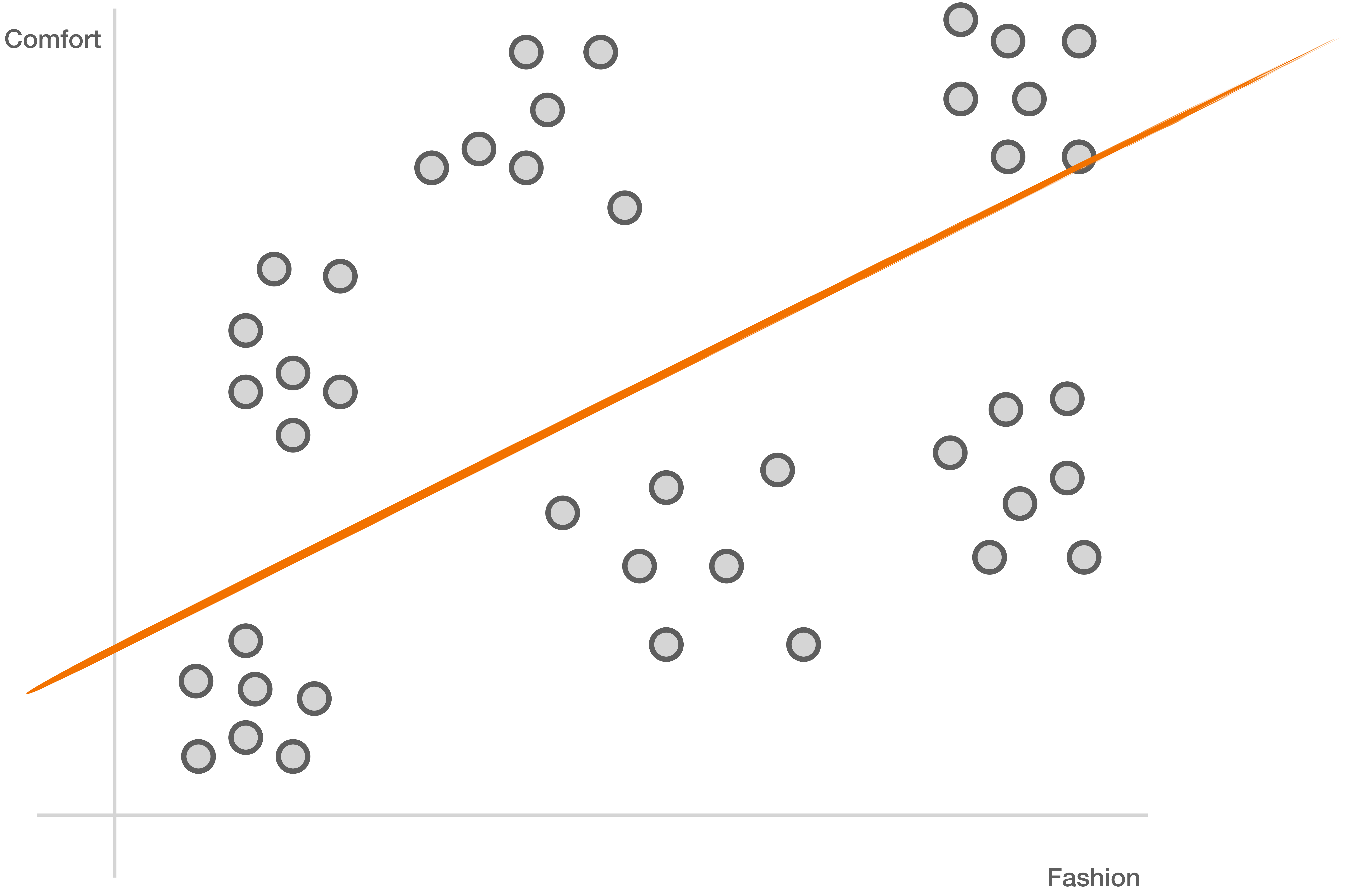
Nope. And that's OK.

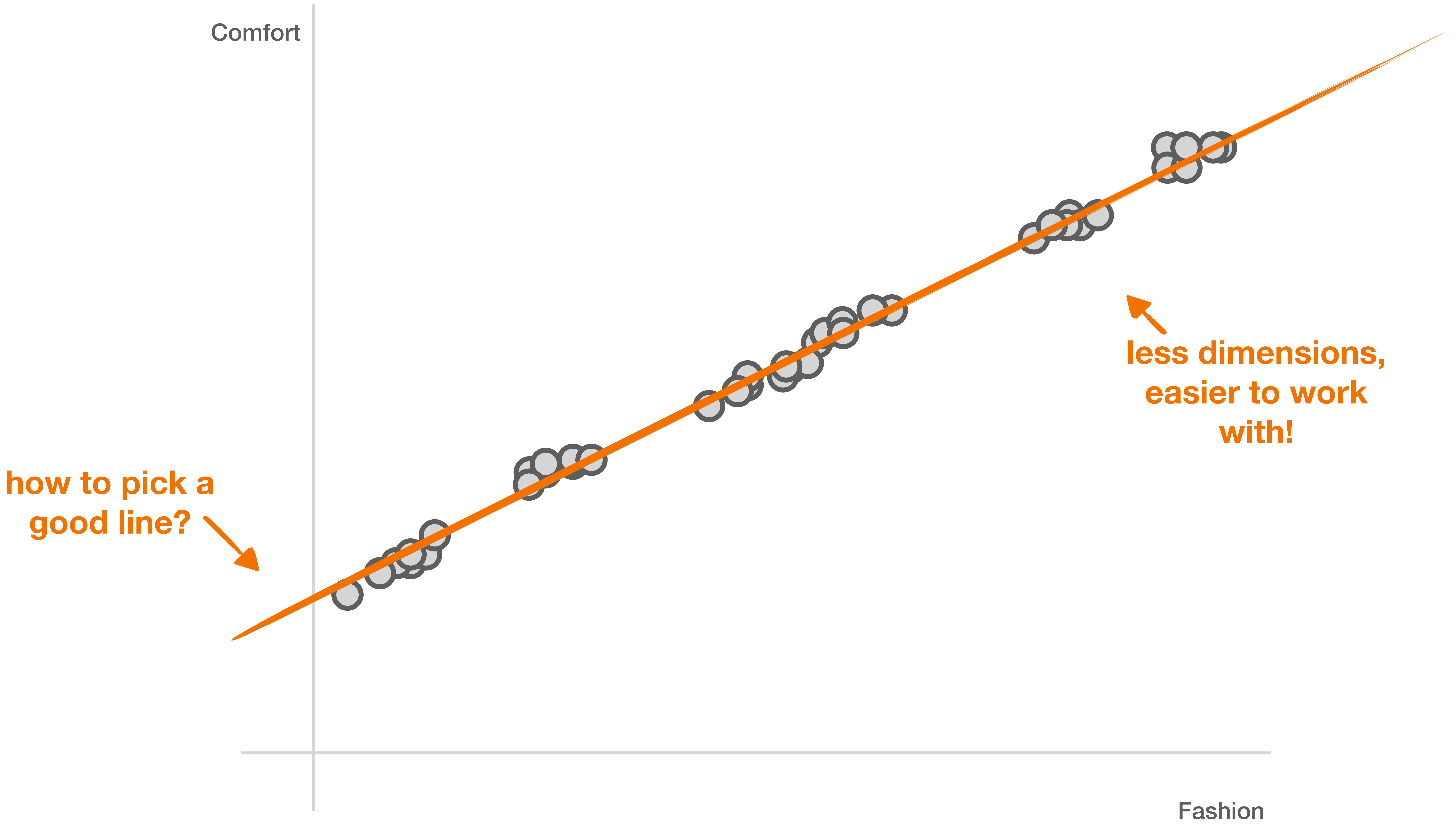
- **K-means** is an *indeterministic* algorithm—it has built-in randomness

UNSUPERVISED LEARNING

Clustering
Dimension reduction

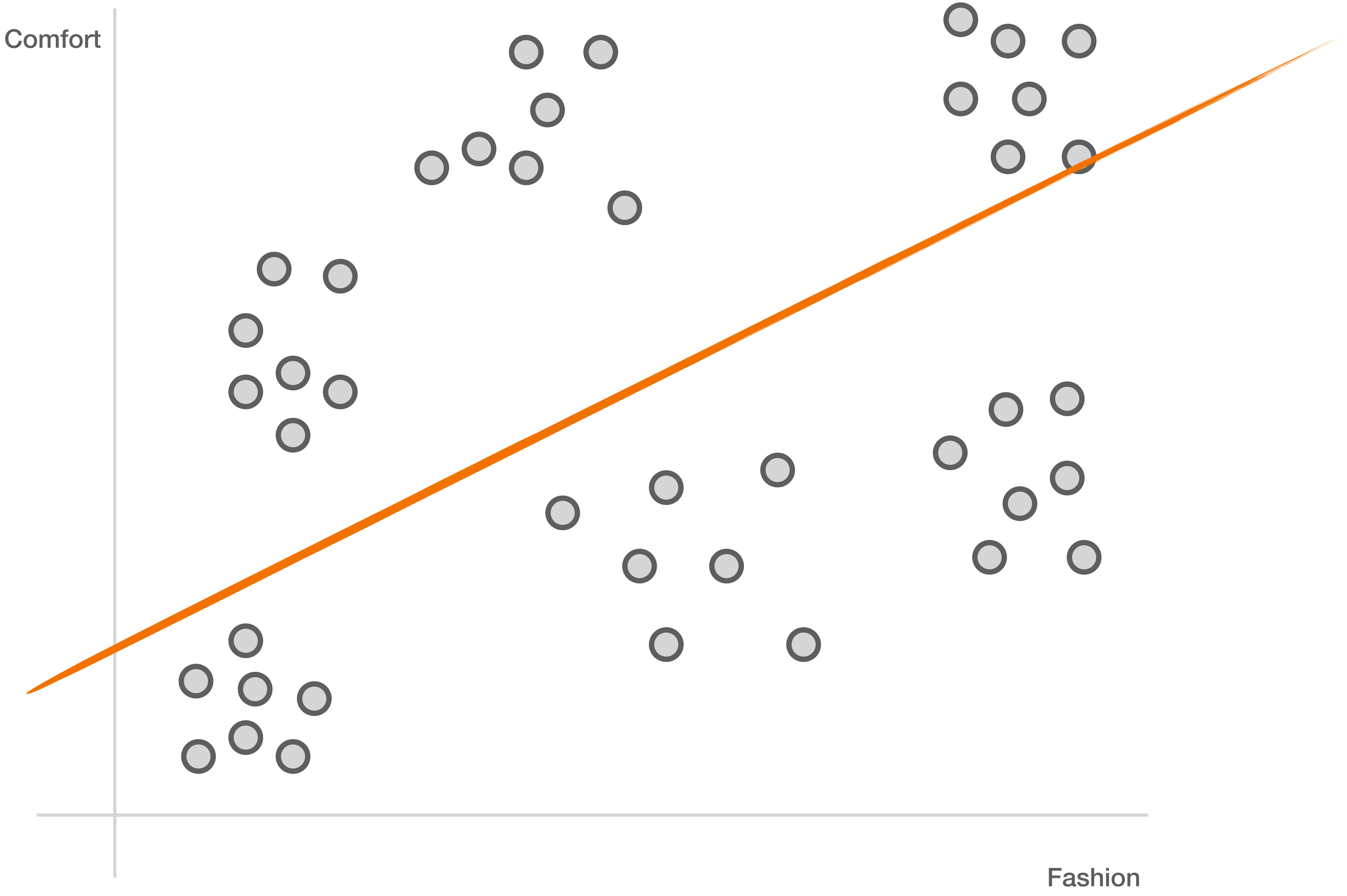
Principle Component Analysis



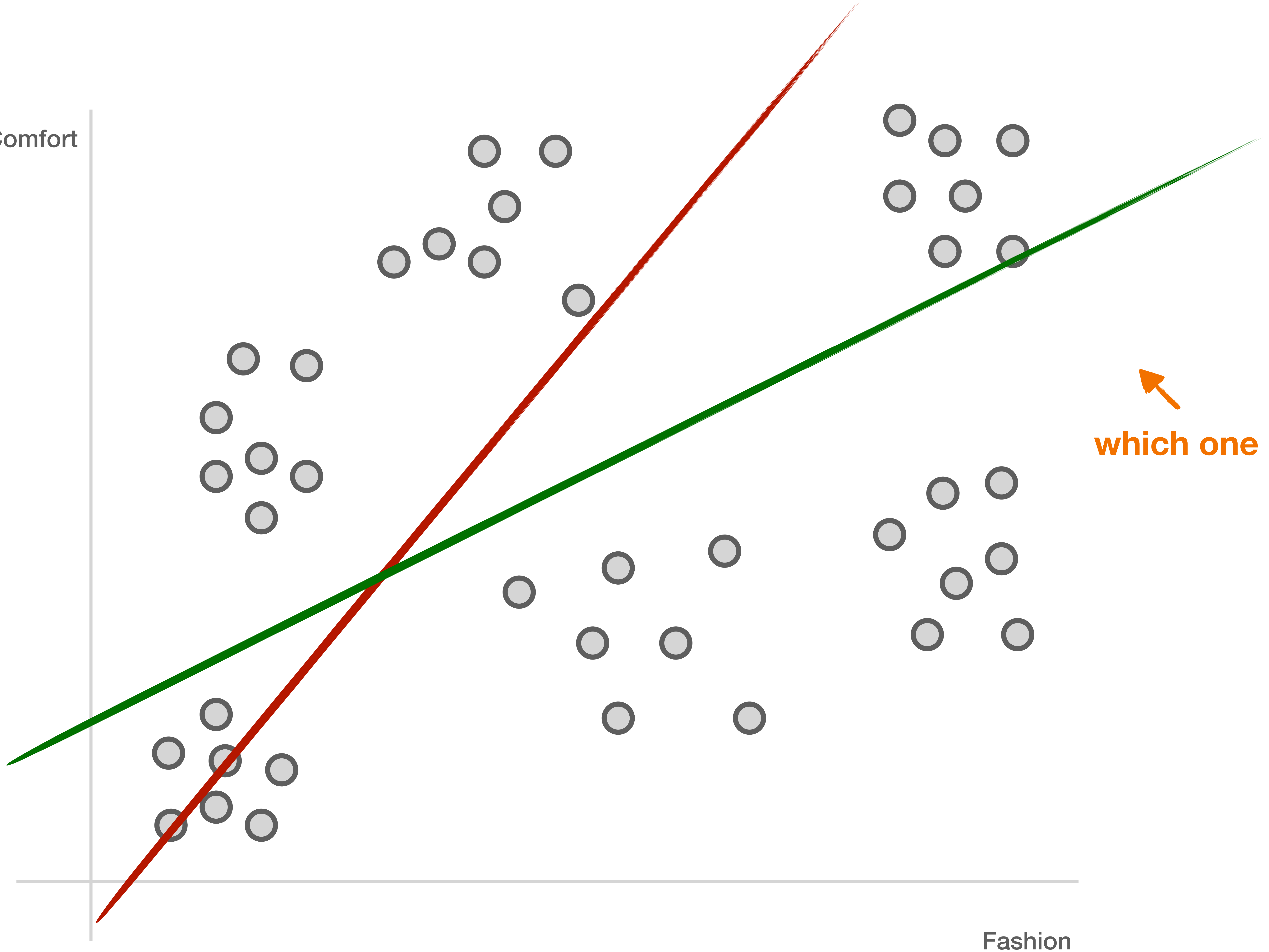


- by **projecting** the samples down to a smaller dimension, they are easier to work with.
  (because the centroids have less “space” to move around)

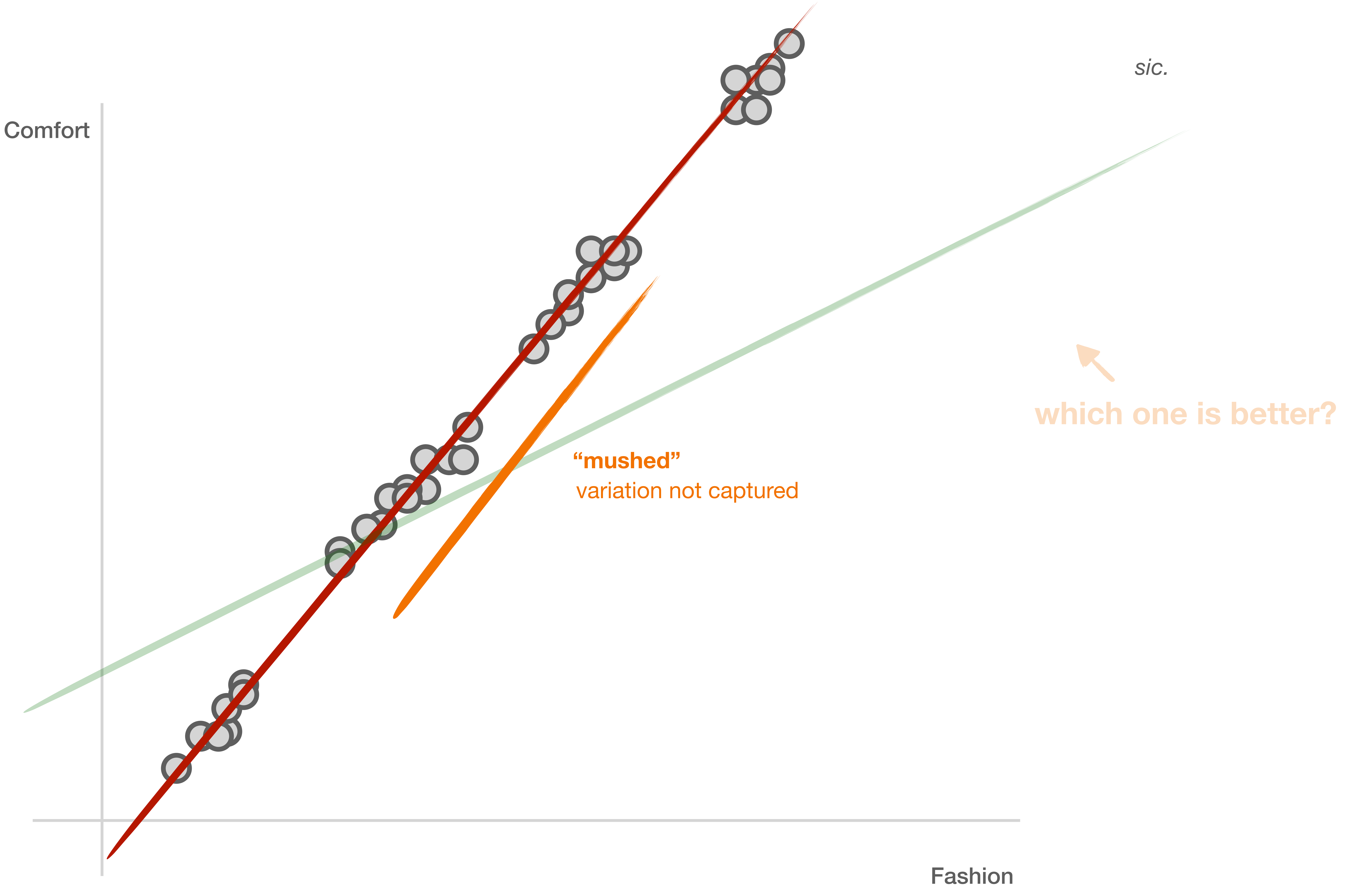
how to pick a good line?



how to pick a good line?



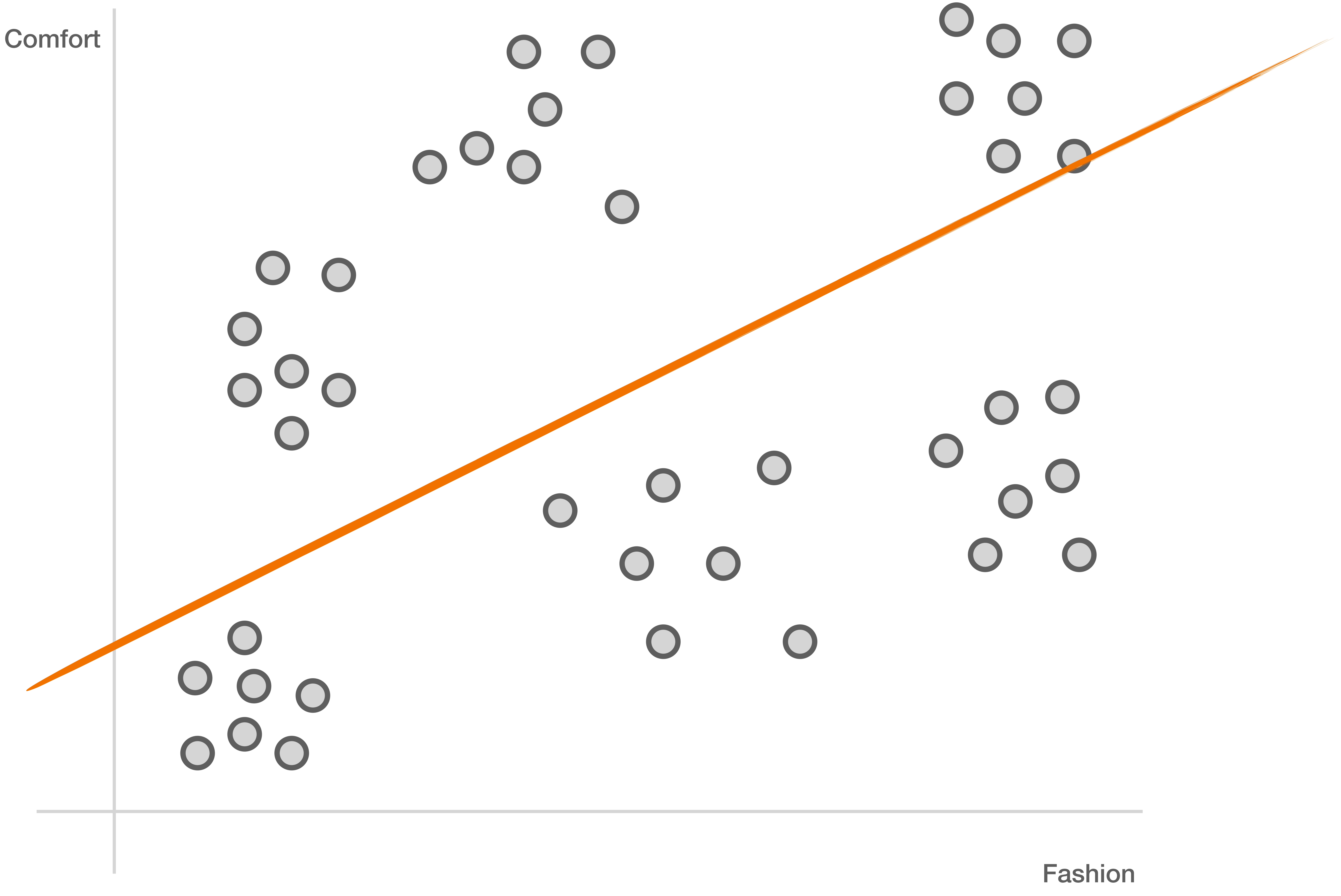
how to pick a good line?



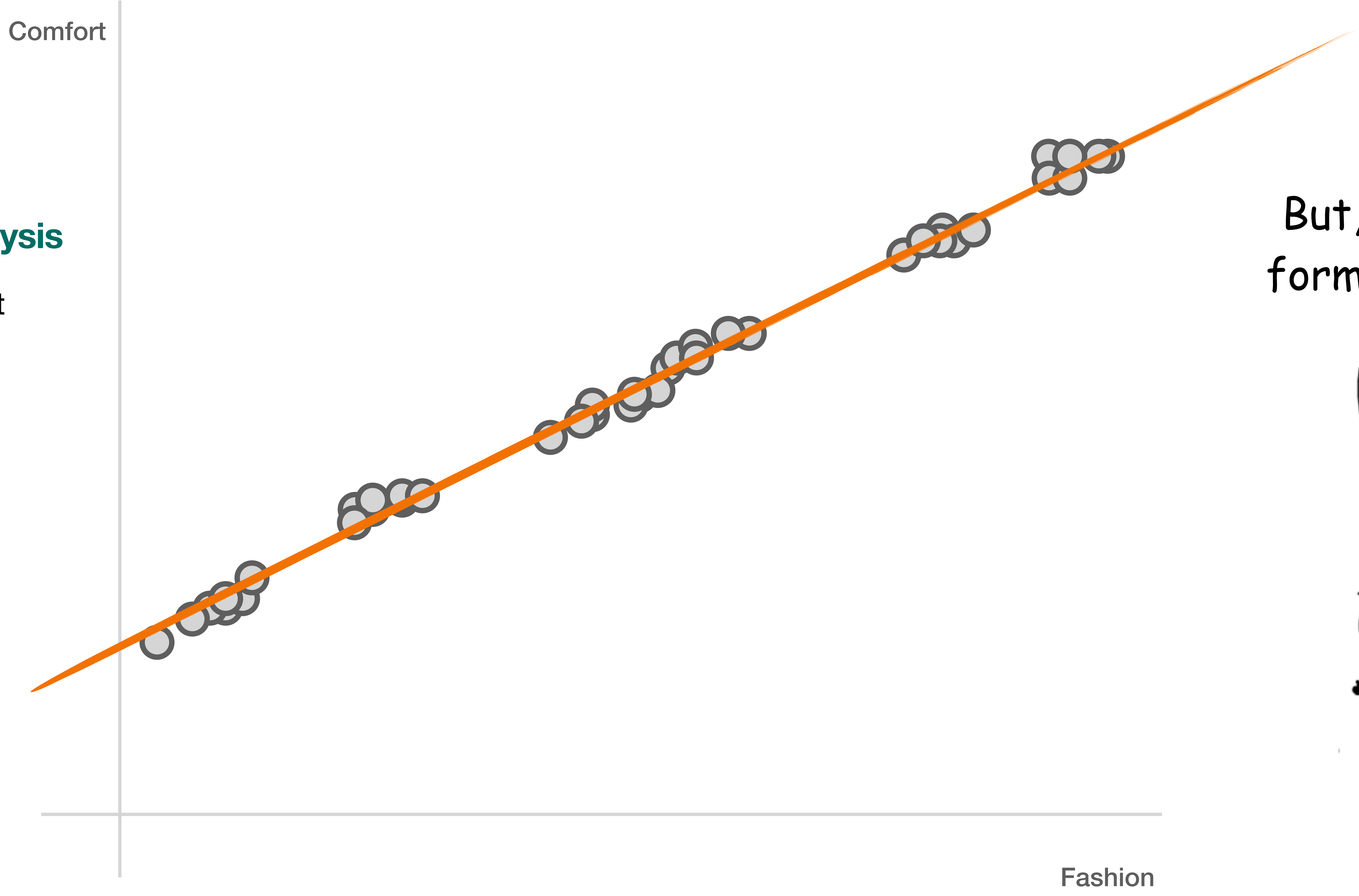
- a good “**projection**” captures the **variation** in the data

how to pick a good line?

- Find a good line (**basis**) that **maximizes variation**

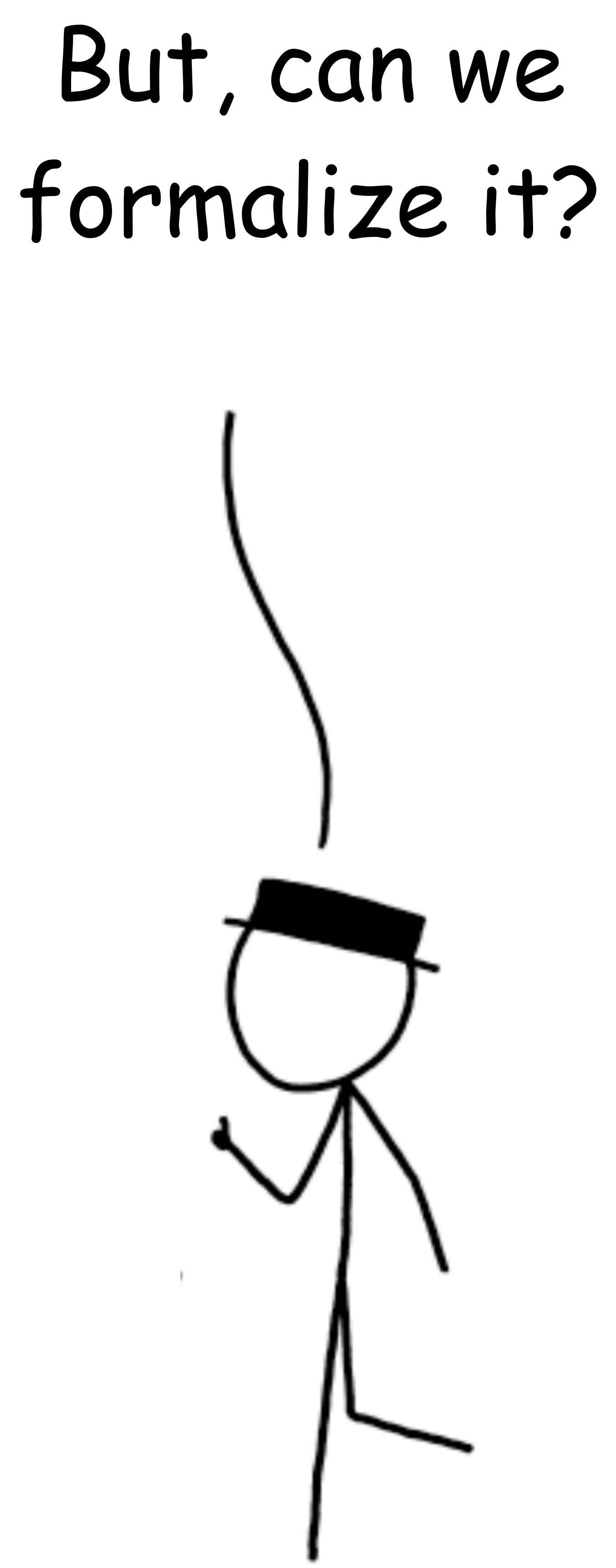


how to pick a good line?



principle component analysis

- Find a good line (**basis**) that **maximizes variation**
- **Project** samples down



Principle Component Analysis

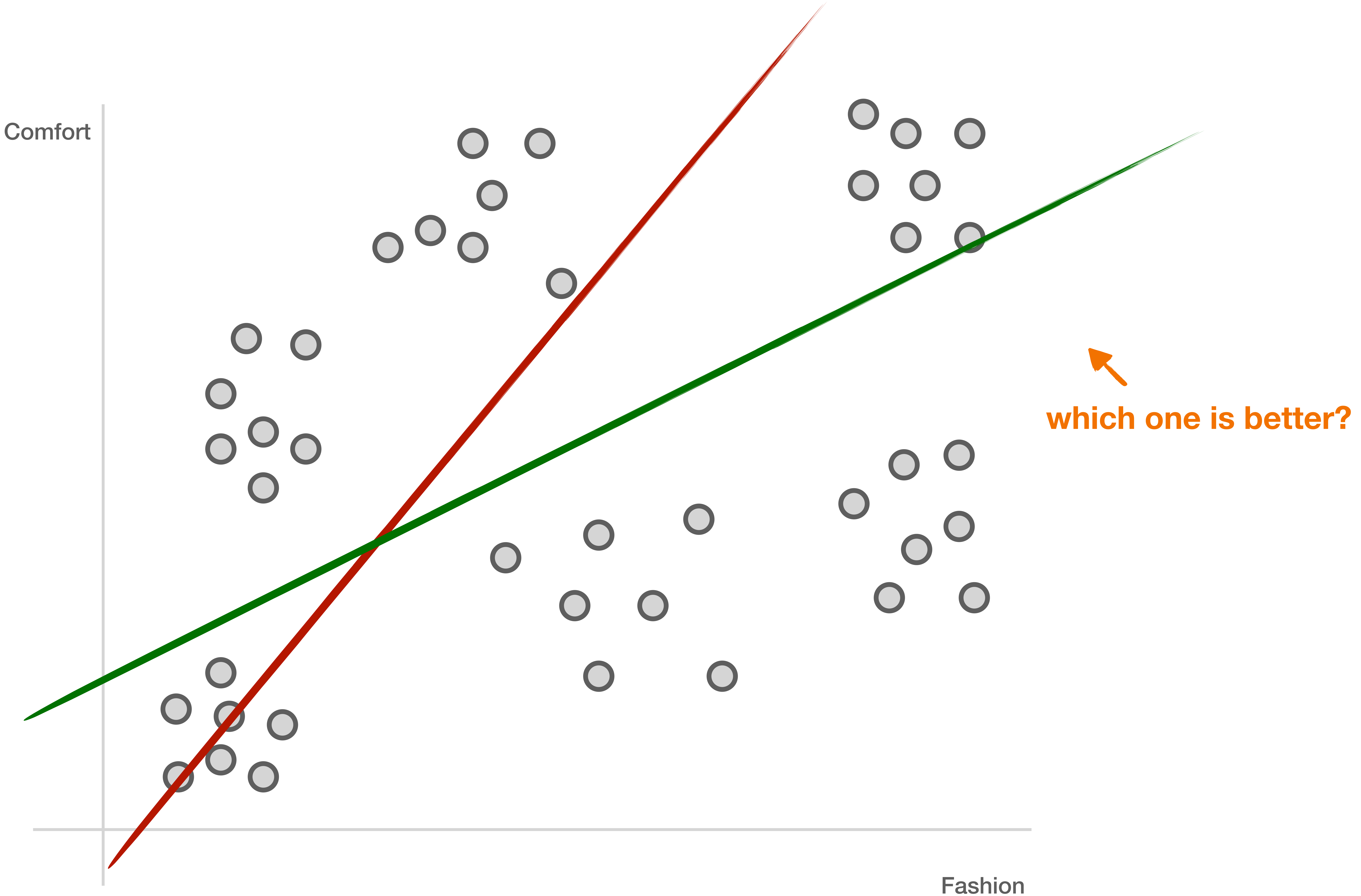
First Component: maximize the explained variance

$$\mathbf{w}_{(1)} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2 \right\}$$

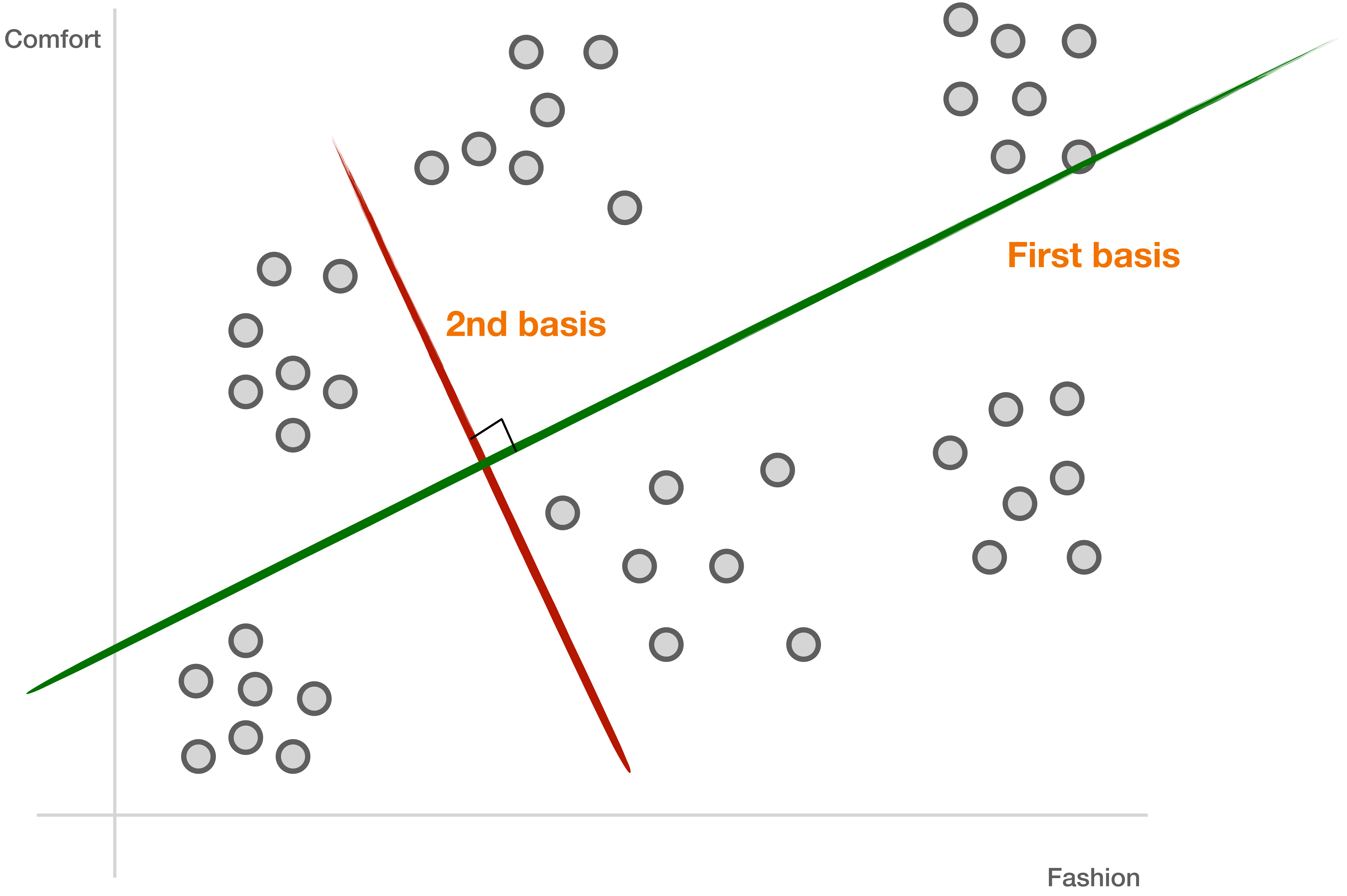
Further Components: maximize the explained variance in remainders

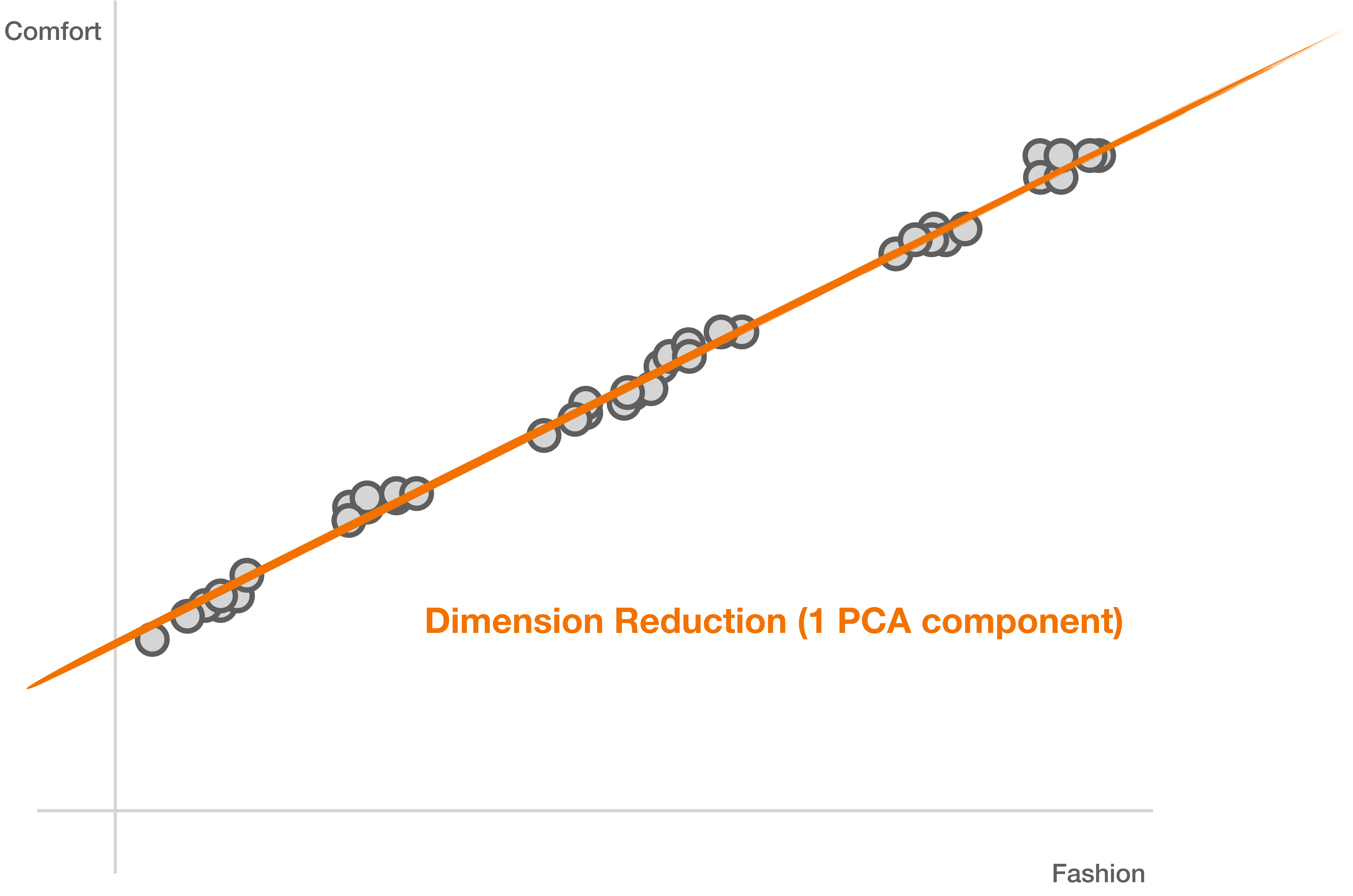
$$\mathbf{w}_{(k)} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max} \left\{ \left\Vert \hat{\mathbf{X}}_k \mathbf{w} \right\Vert^2 \right\} \qquad \hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{w}_{(s)}\mathbf{w}_{(s)}^{\mathsf{T}}$$

First Component



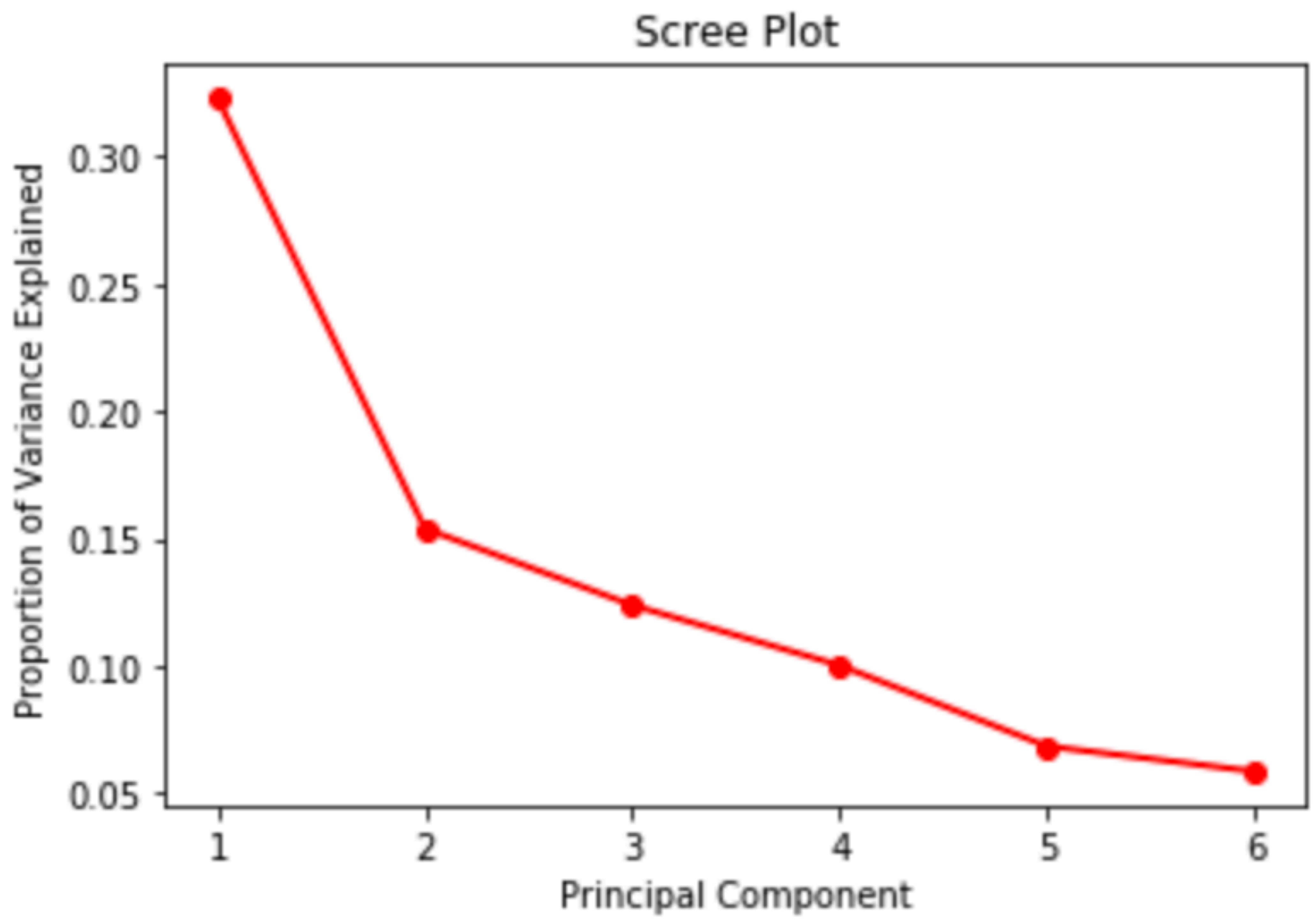
PCA




Comfort
Dimension Reduction (1 PCA component)
Fashion

- Linear transformation of original features
- Dimension reduction
- Compression
- Denoise
- Lose interpretability

- How to choose k (hyperparamter)



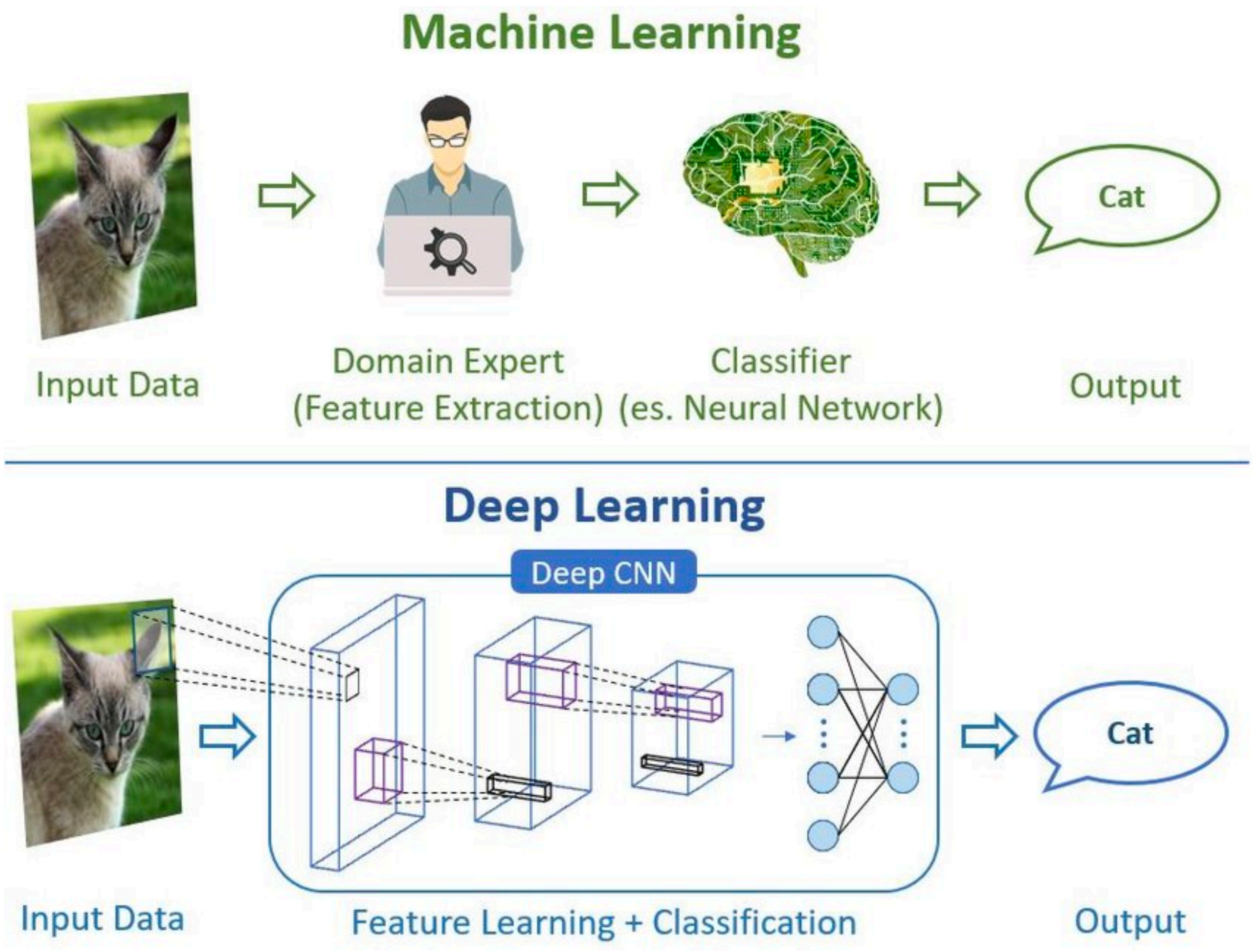
POTENTIAL PITFALLS

Things that can go wrong

- Inconsistent preprocessing (e.g., different scaling/normalization)
- Data leakage (e.g., temporal or mixing subjects)

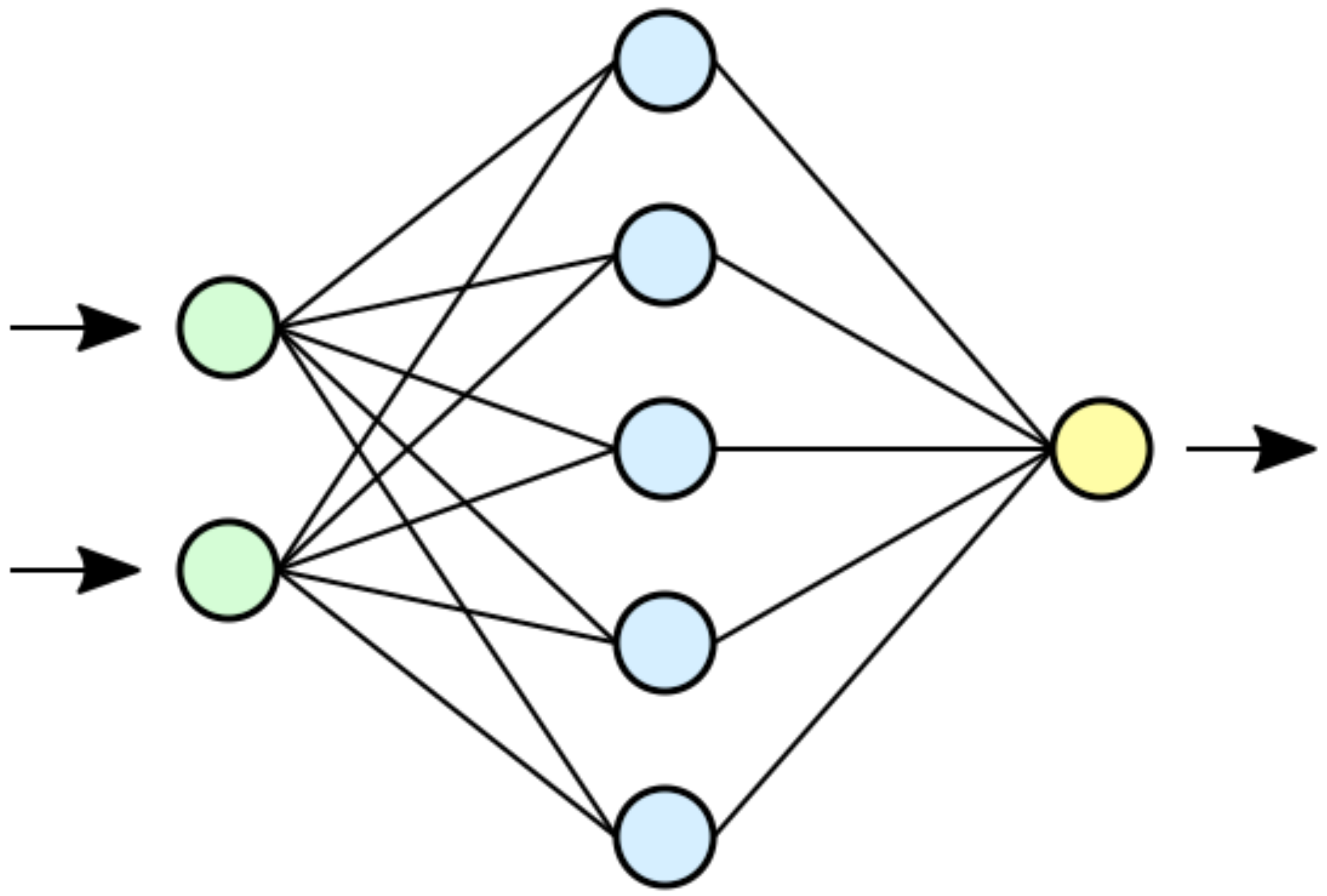
- Model is used on test data that has changed
- Selecting appropriate metrics (e.g., is 99% accuracy good enough?)
- Hidden confounders (e.g., golf is correlated with heart attacks)
- Spurious correlations (e.g., hospital ID on images)
- Performance on subgroups may be missing

Classic vs. Deep ML



Cool, so…
what’s next?

Cool, so… what's next?



Neural Networks!

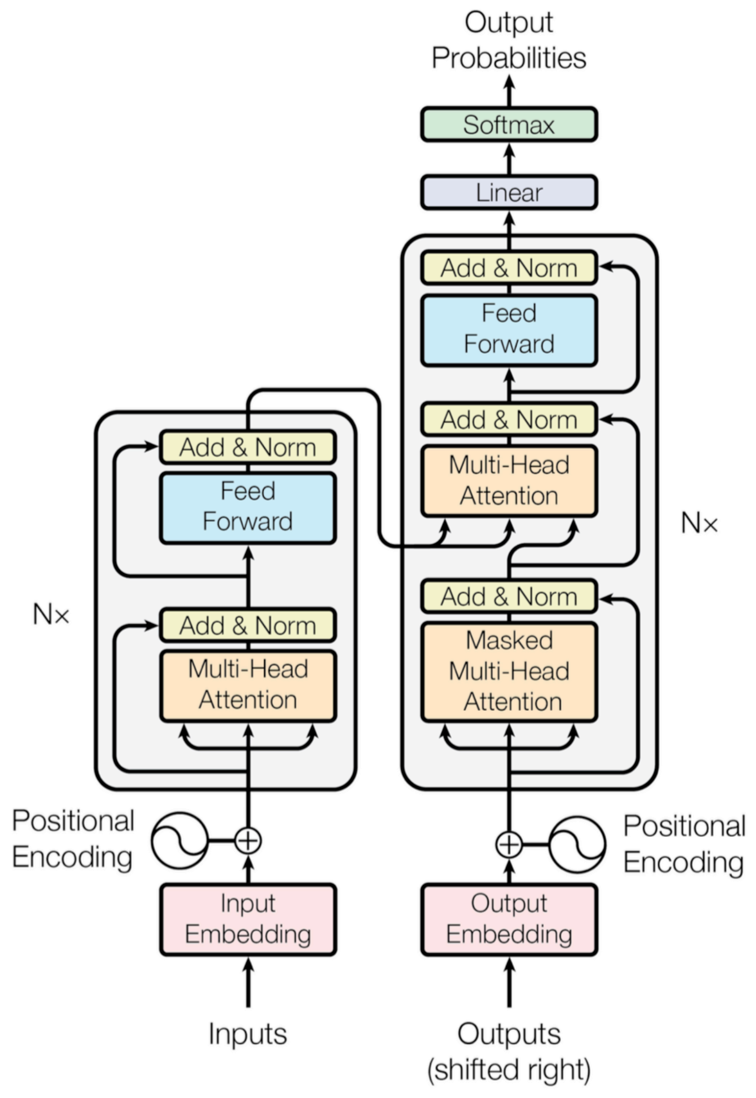
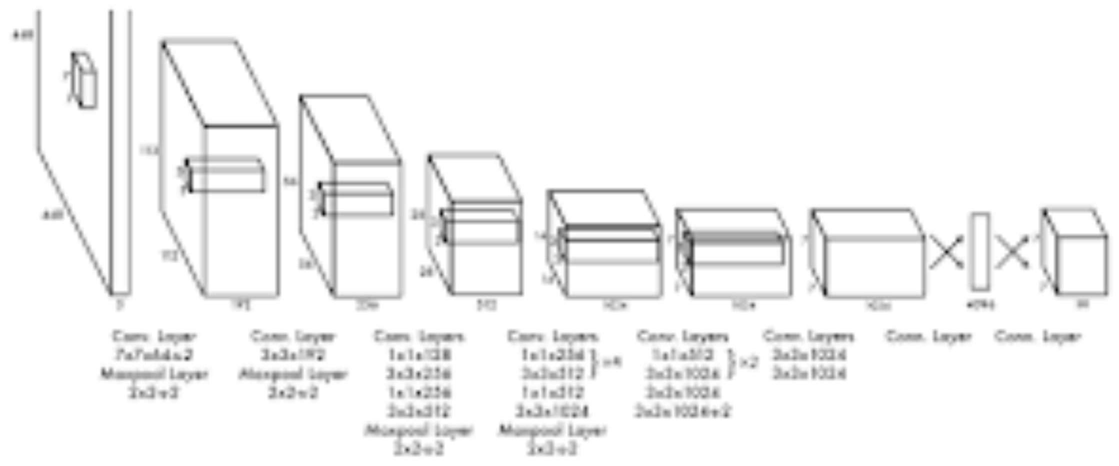


Figure 1: The Transformer - model architecture.

Natural Language Processing



Computer Vision

ENSEMBLE

Wisdom of the crowd

- Guessing the weight of a steer (Sir Francis Calton)
- Key components:
  - Base models with **diversity**
  - Infusion algorithms to integrate base models
- Bagging
- Boosting

Bagging

- Build several instances of a classifier using a subset of the original training data
- Aggregate (Averaging) the results
- Reduce variance
- Ex: random forest

- https://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator

Boosting

- Fit a sequence of weak learners on repeatedly modified versions of the data.
- A weighted majority vote
- Ex: Adaboost, Gradient boosting

SELF-SUPERVISED LEARNING

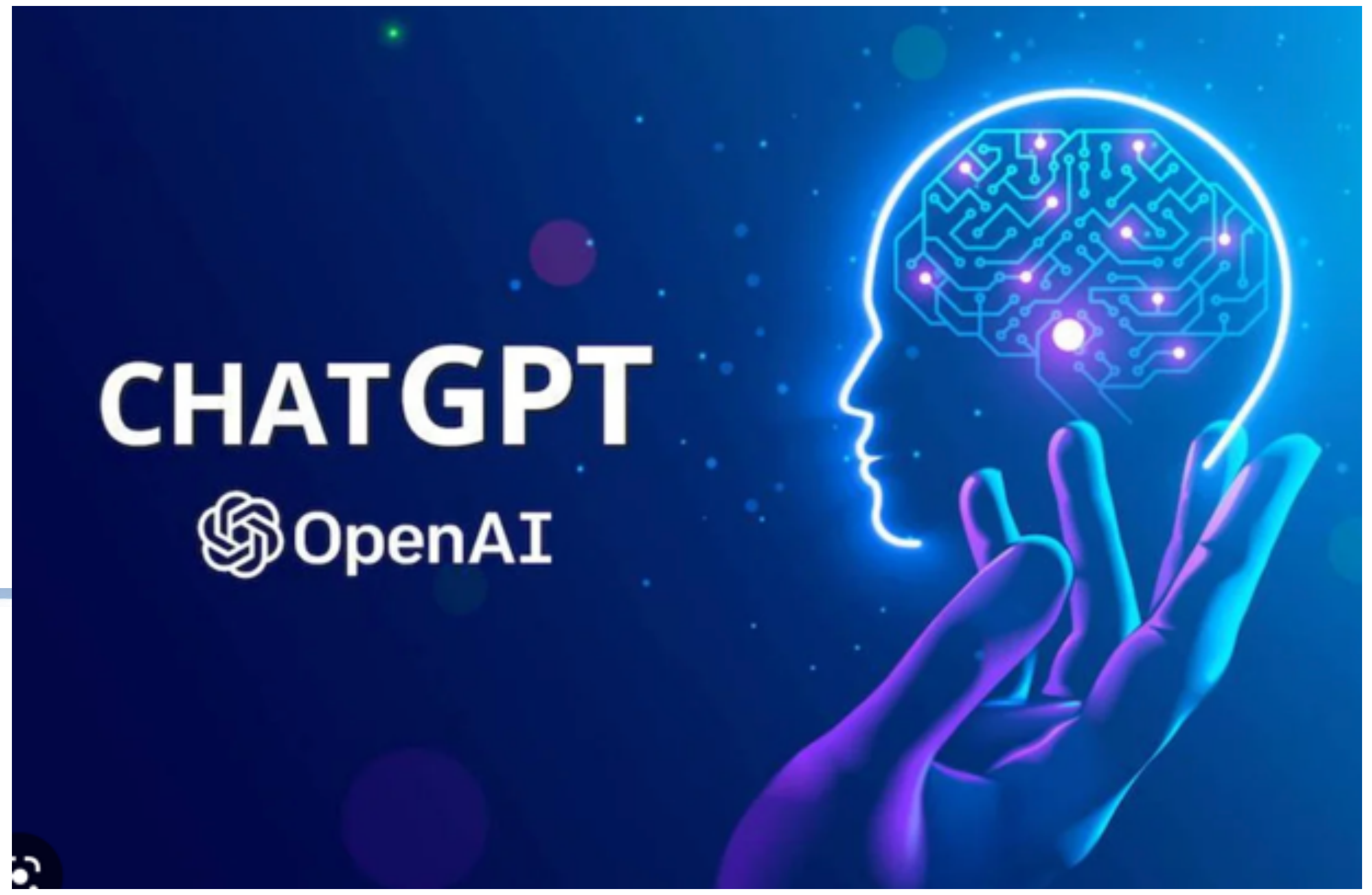
Exploiting Unlabeled Data

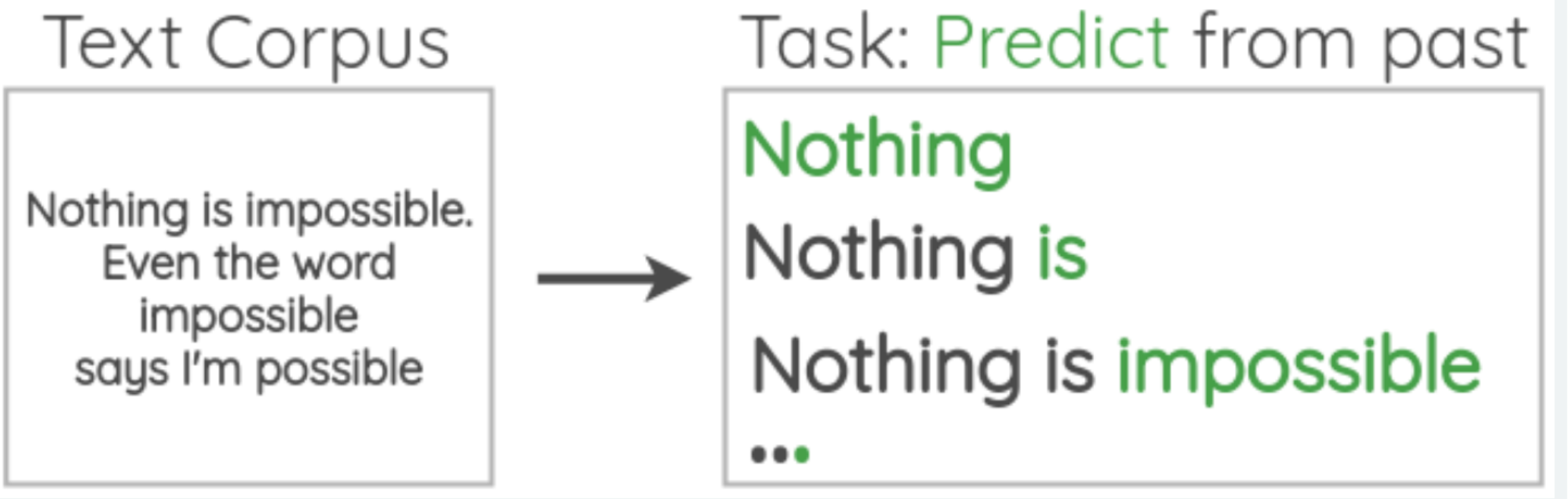
A lot of unlabeled data is plentiful and cheap, e.g.,

- Document off the web
- Speech samples
- Images and video

But labeling can be expensive

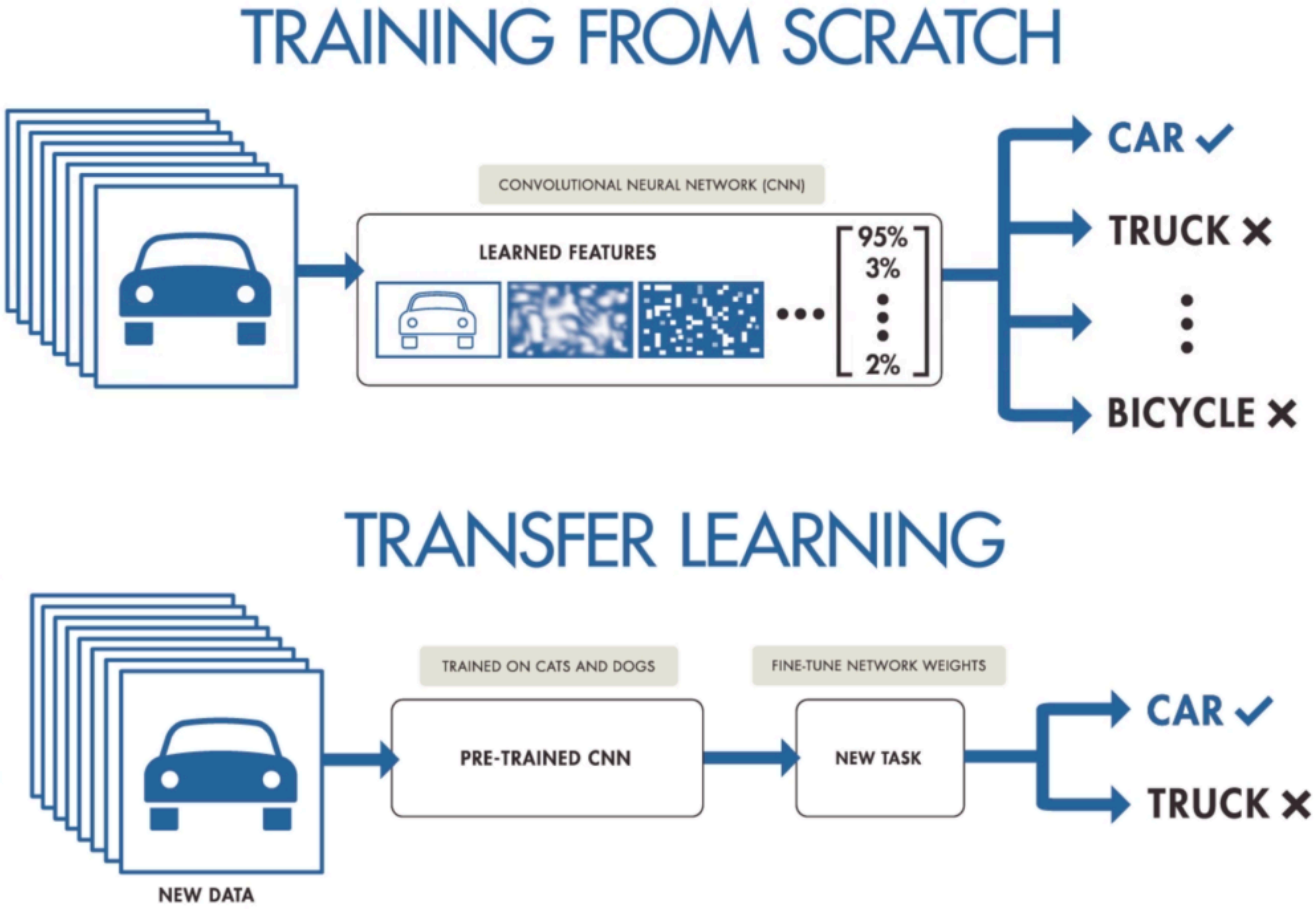
Self-supervised learning



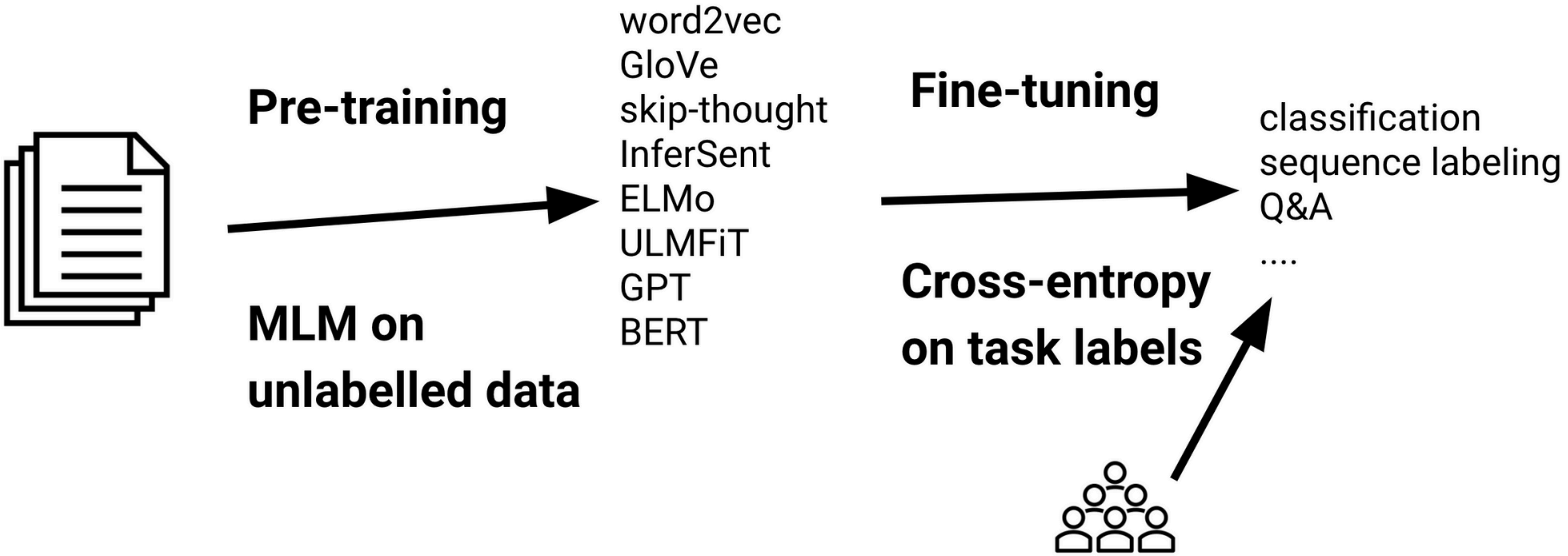


A critical step in training large language models

TRANSFER LEARNING



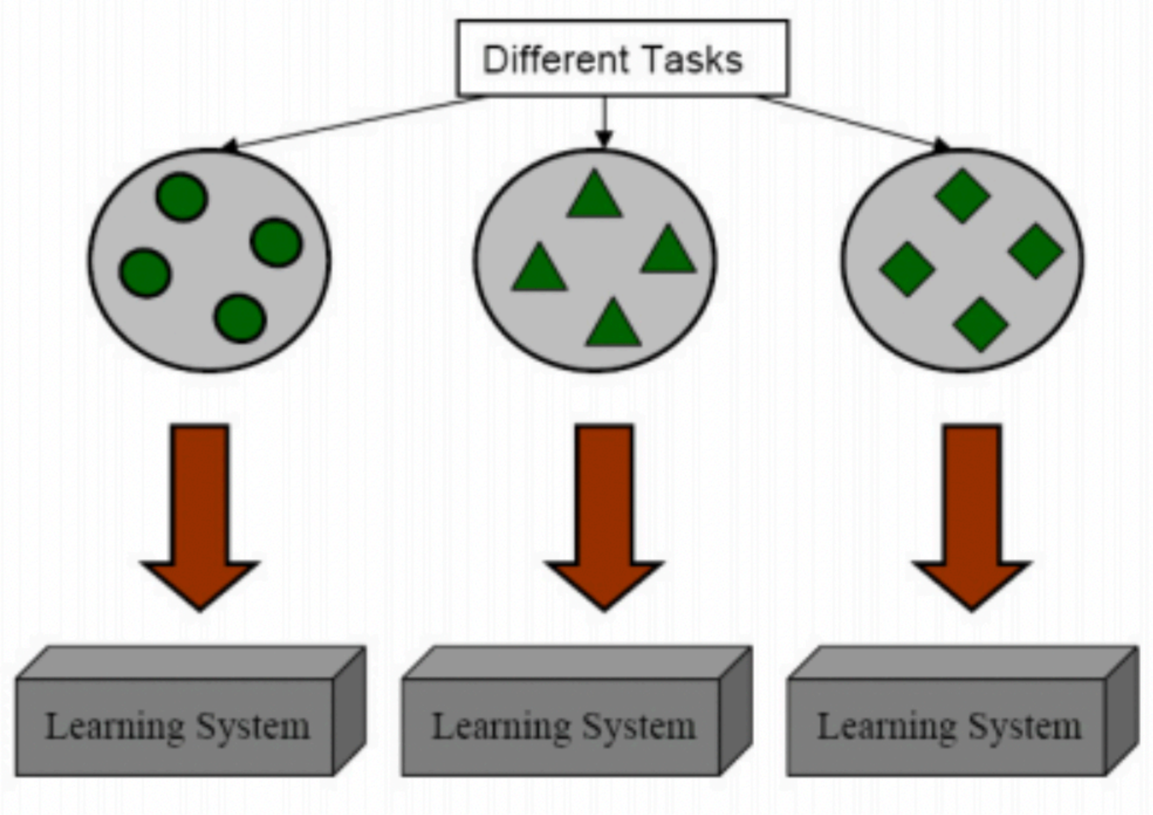
TRAINING FROM SCRATCH
CONVOLUTIONAL NEURAL NETWORK (CNN)
LEARNED FEATURES
95%
3%
2%
CAR
TRUCK
BICYCLE
TRANSFER LEARNING
TRAINED ON CATS AND DOGS
FINE-TUNE NETWORK WEIGHTS
PRE-TRAINED CNN
NEW TASK
CAR
TRUCK
NEW DATA



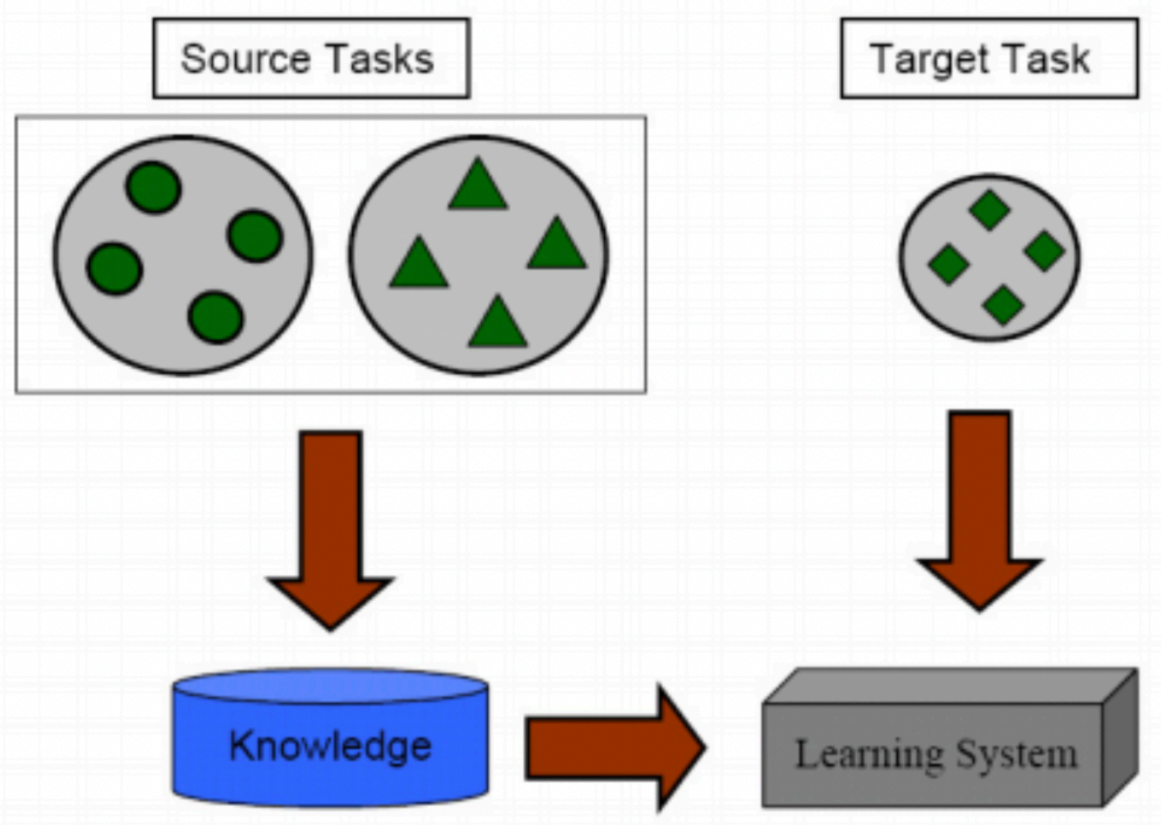
Pre-training
MLM on
unlabelled data
word2vec
GloVe
skip-thought
InferSent
ELMo
ULMFiT
GPT
BERT
Fine-tuning
Cross-entropy
on task labels
classification
sequence labeling
Q&A
....

Transfer learning and multi-task learning

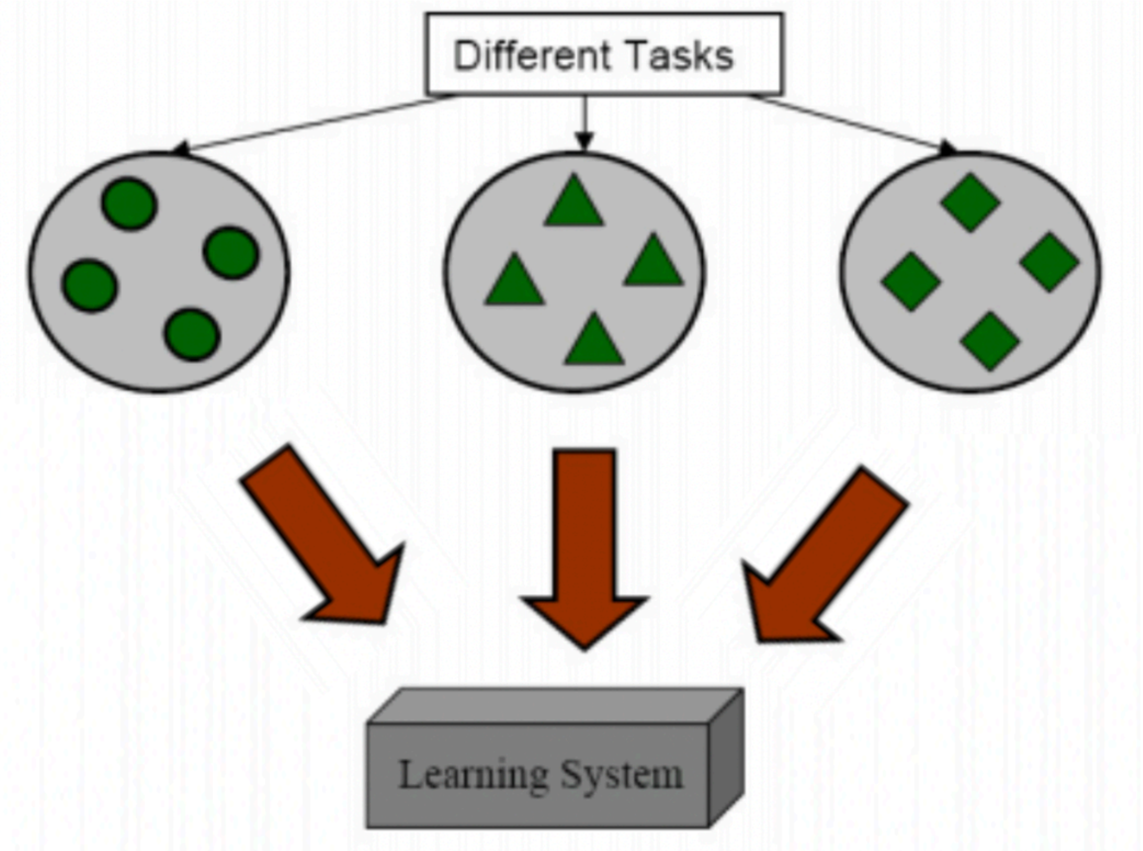
Traditional machine learning



Transfer learning



Multi-task learning



ACTIVE LEARNING

Active Learning, aka Query Learning

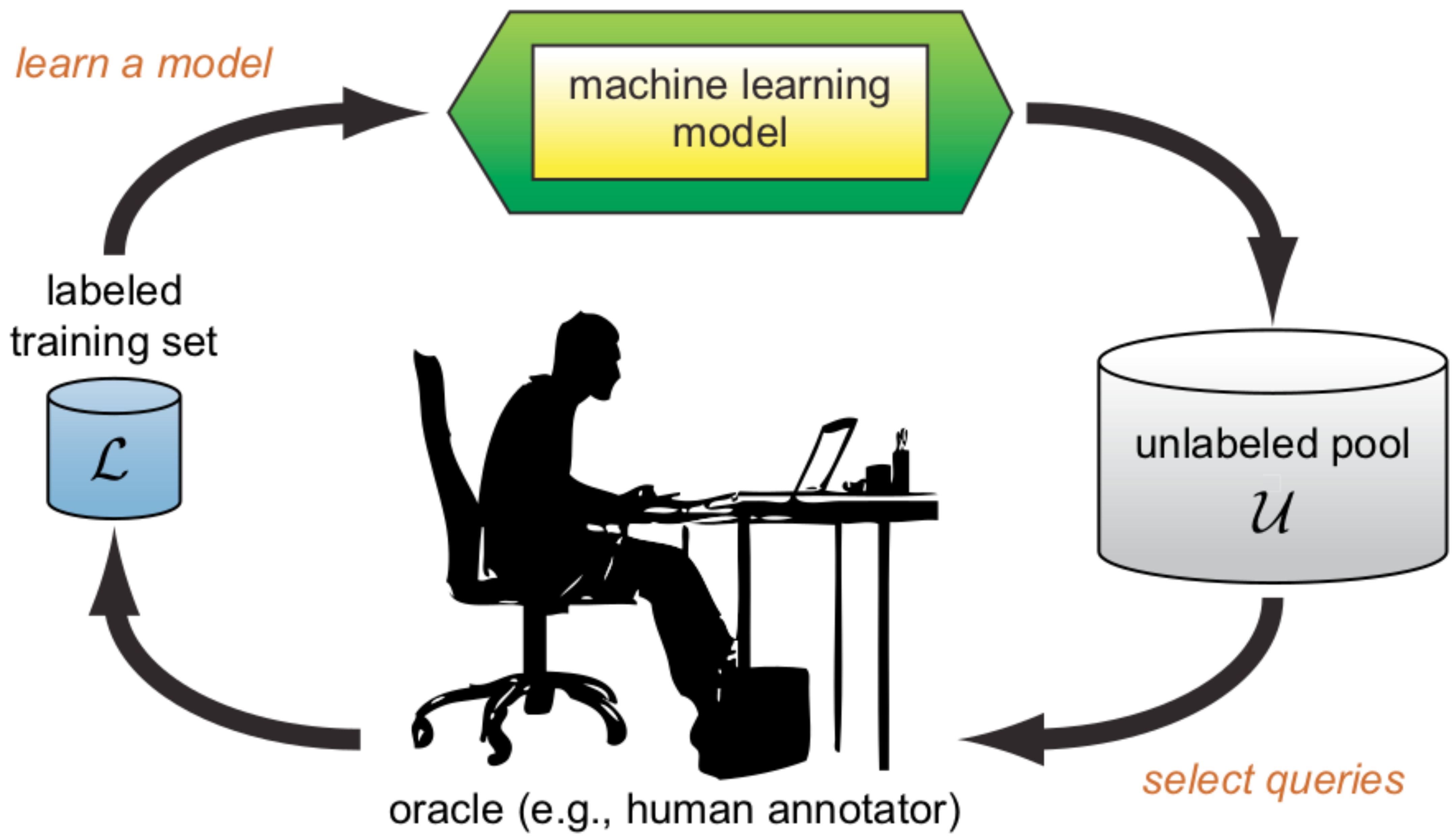


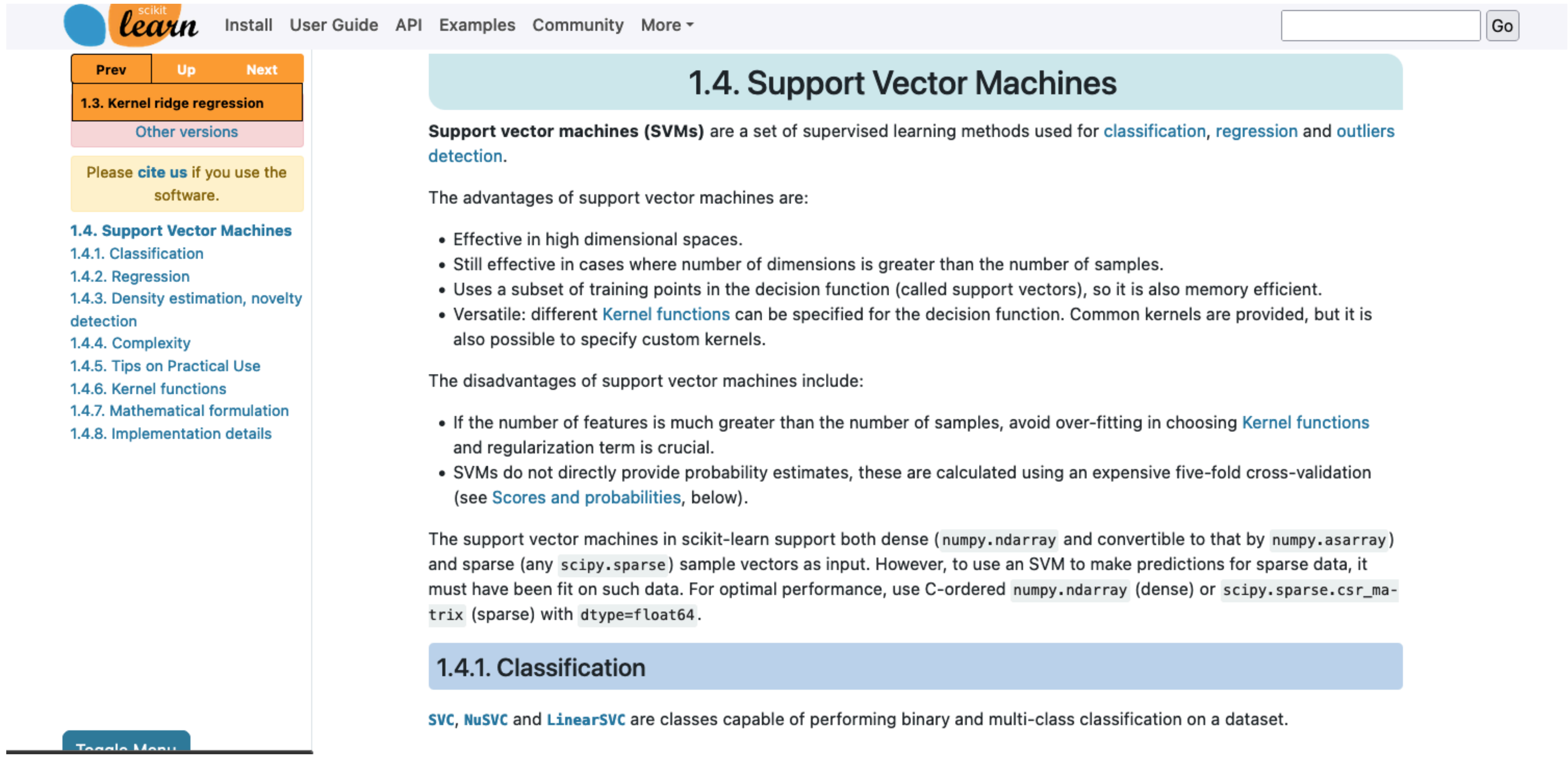
Fig. The active learning cycle.

Repeat

1. Choose unlabeled sample
2. Annotate the chosen unlabeled sample
3. The model trains on the labeled data set

Cheap unlabeled data
Expensive labeled data

Cool, so… what’s next?


Install User Guide API Examples Community More

Prev Up Next

1.3. Kernel ridge regression

Other versions

Please **cite us** if you use the software.

1.4. Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

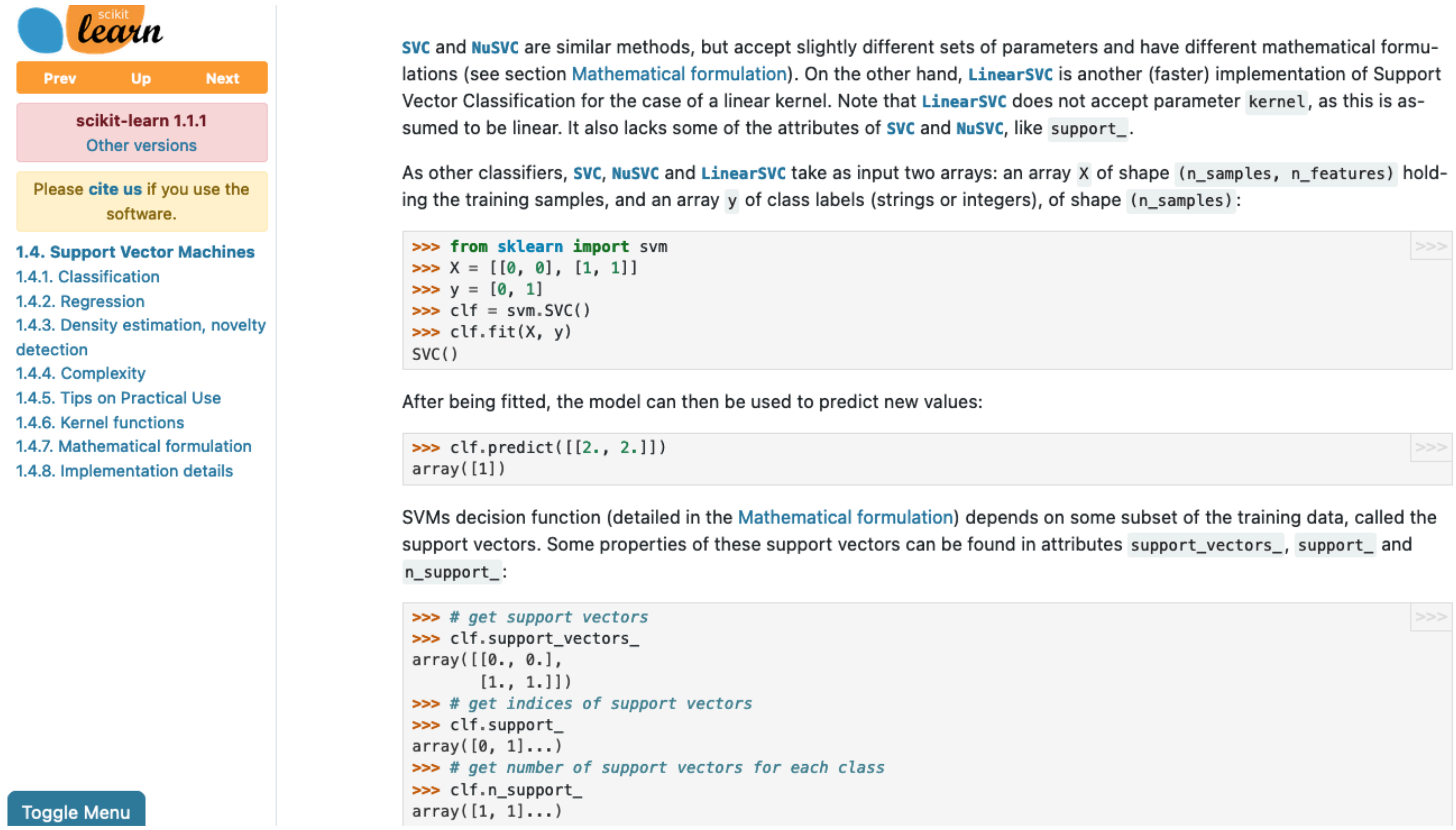
- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

The support vector machines in scikit-learn support both dense (`numpy.ndarray` and convertible to that by `numpy.asarray`) and sparse (any `scipy.sparse`) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered `numpy.ndarray` (dense) or `scipy.sparse.csr_matrix` (sparse) with `dtype=float64`.

1.4.1. Classification

`SVC`, `NuSVC` and `LinearSVC` are classes capable of performing binary and multi-class classification on a dataset.

Toggle Menu


Prev Up Next

scikit-learn 1.1.1
Other versions

Please **cite us** if you use the software.

Toggle Menu

`SVC` and `NuSVC` are similar methods, but accept slightly different sets of parameters and have different mathematical formulations (see section Mathematical formulation). On the other hand, `LinearSVC` is another (faster) implementation of Support Vector Classification for the case of a linear kernel. Note that `LinearSVC` does not accept parameter `kernel`, as this is assumed to be linear. It also lacks some of the attributes of `SVC` and `NuSVC`, like `support_`.

As other classifiers, `SVC`, `NuSVC` and `LinearSVC` take as input two arrays: an array `X` of shape `(n_samples, n_features)` holding the training samples, and an array `y` of class labels (strings or integers), of shape `(n_samples)`:

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC()
```

After being fitted, the model can then be used to predict new values:

```
>>> clf.predict([[2., 2.]])
array([1])
```

SVMs decision function (detailed in the Mathematical formulation) depends on some subset of the training data, called the support vectors. Some properties of these support vectors can be found in attributes `support_vectors_`, `support_` and `n_support_`:

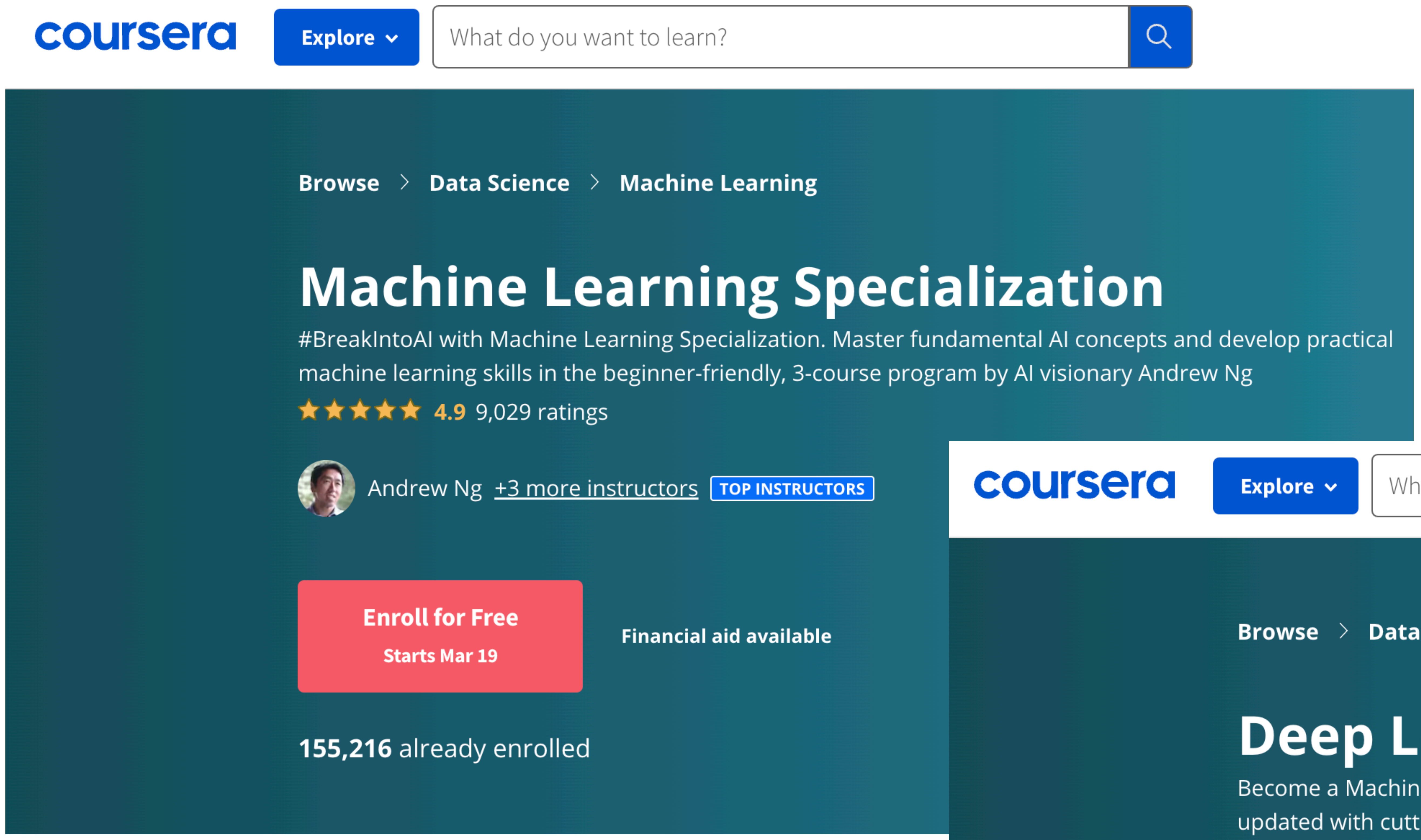
```
>>> # get support vectors
>>> clf.support_vectors_
array([[0., 0.],
       [1., 1.]])
>>> # get indices of support vectors
>>> clf.support_
array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_
array([1, 1]...)
```

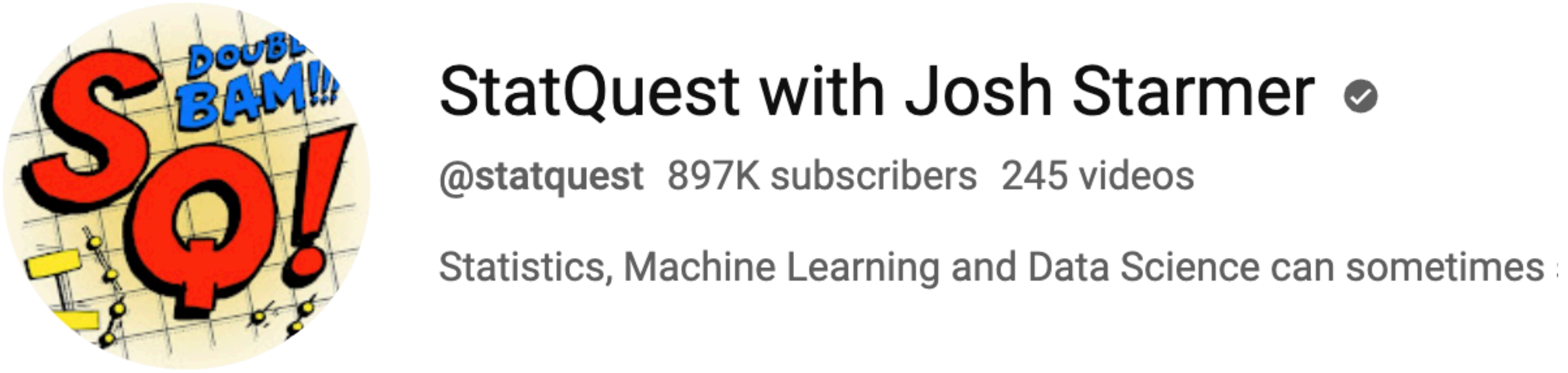
no seriously,
learn to read
docs

scikit-learn.org

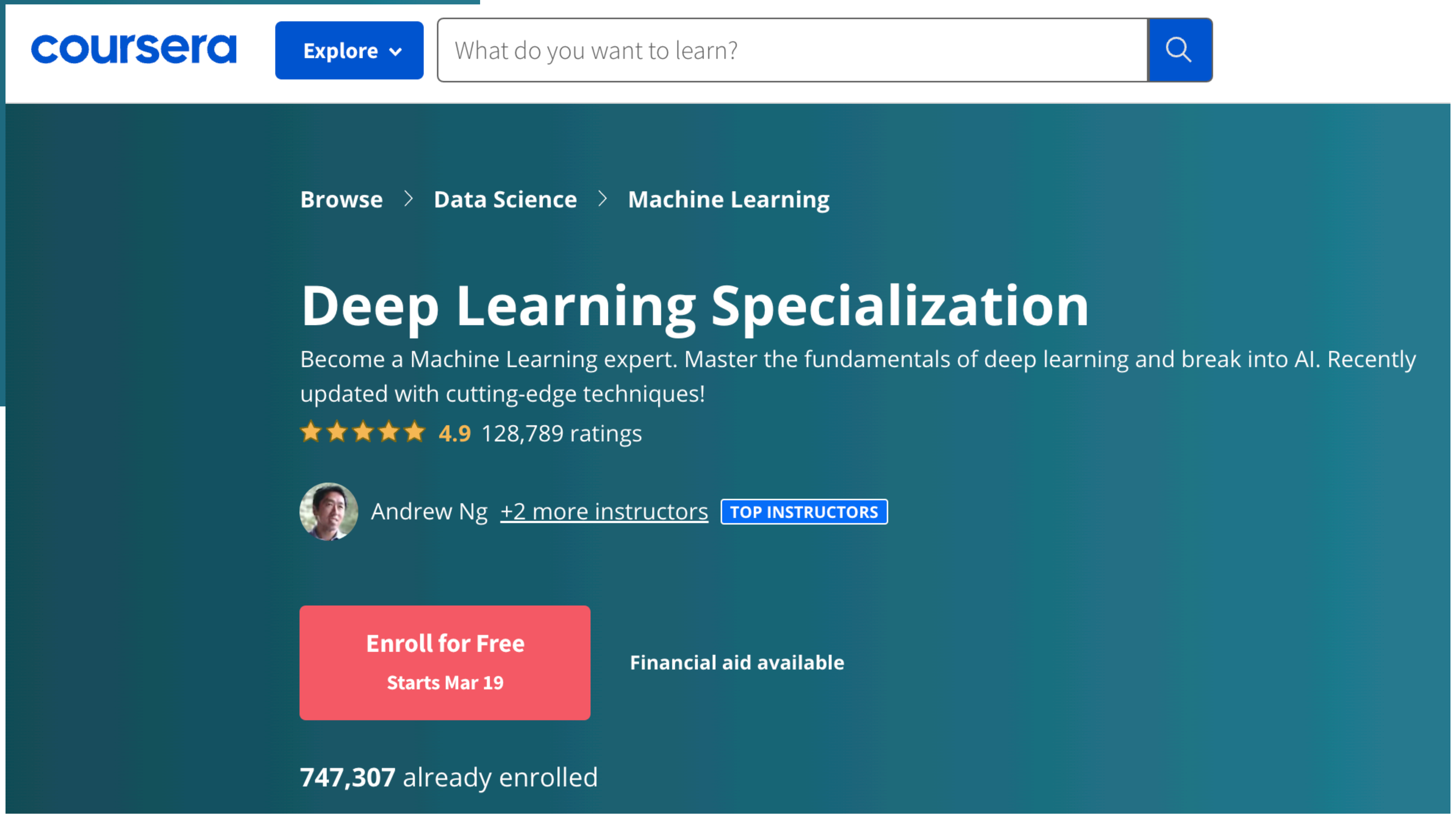
…and general google-fu!

Cool, so… what's next?





towards
data science



(and have lots of fun!)

Ethics

- Interpretability of the model
- Security
- Privacy
- Trustworthiness
- Biases
- Fairness
- Socioeconomic consequences
- Unintended consequences
- Trolly problem

Thank you.

AIBridge

Samuel Ren, Henry M Gunn High School

Jiaming Situ, Homestead High School

Houjun Liu, The Nueva School

Xin Liu, Professor, Computer Science. UC Davis

With acknowledgments to

AI Institute for Food Systems

Saratoga Public Library